SOME ARTIFICIAL NEURAL NETWORK RULE EXTRACTION ALGORITHMS FOR NONLINEAR REGRESSION PROBLEMS AND THEIR APPLICATIONS

A Thesis
Submitted to the Faculty of Graduate Studies and Research
In partial Fulfillment of the Requirements
For the Degree of
Doctor of Philosophy
in
Software Systems Engineering
University of Regina

By
Ka Hei Veronica Chan
Regina, Saskatchewan
July, 2017

Copyright 2017: K. H. V. Chan
Ka Hei Veronica Chan, candidate for the degree of Doctor of Philosophy in Software Systems Engineering, has presented a thesis titled, *Some Artificial Neural Network Rule Extraction Algorithms For Nonlinear Regression Problems And Their Applications*, in an oral examination held on July 17, 2017. The following committee members have found the thesis acceptable in form and content, and that the candidate demonstrated satisfactory knowledge of the subject material.

External Examiner: *Dr. Weichang Du, University of New Brunswick*

Supervisor: Dr. Christine Chan, Software Systems Engineering

Committee Member: Dr. Craig Gelowitz, Software Systems Engineering

Committee Member: Dr. Mohamed El-Darieby, Software Systems Engineering

Committee Member: Dr. Paitoon Tontiwachwuthikul, Industrial and Process Systems Engineering

Committee Member: *Dr. Andrei Volodin, Department of Mathematics and Statistics*

Chair of Defense: Dr. Harold Weger, Department of Biology

*via SKYPE

**Not present at defense
ABSTRACT
Inspired by the operation mechanism of the human brain, artificial neural network (ANN) is a computational system that exhibits high predictive power, great adaptability, and robustness. However, despite the high predictive accuracy, ANN models are opaque and do not provide researchers with explicit relationships between the predictor and predicted attributes of the studied problem. Therefore, to overcome this drawback, three ANN rule extraction algorithms, namely the PWL-ANN, the enhanced-PWL-ANN, and the EURECA algorithms, were proposed in this study. The PWL-ANN algorithm extracts rules from trained ANN models by approximating the hidden neuron sigmoid functions by 3-piece linear functions. Whereas, the enhanced-PWL-ANN algorithm extracts rules by approximating the hidden neuron sigmoid function by some (N+1)-piece linear functions. The enhanced algorithm also included a (i) sampling algorithm for reducing computation time, and (ii) an sensitivity analysis, for balancing the trade-off between fidelity and comprehensibility of the generated rule sets. The results showed that by increasing the number of breakpoints in the piece-wise linear functions to more than 2, the accuracy and fidelity of the generated rules were improved. However, two drawbacks of the enhanced-PWL-ANN algorithm were that (1) the algorithm produced output equations that depend on the dataset used to train the original ANN models, and required extra steps to translate the values and equations to the original values, and (2) the accuracies of the generated rules were not high enough to solve the back calculation problems. To overcome these problems, the EURECA algorithm was developed. The algorithm extracts rules from a trained ANN model by first clustering the data according to the hidden weights of the trained ANN, then it models each cluster of the dataset with
multiple linear regression using the real values. The results showed that the EURECA algorithm generated models with higher prediction accuracies than the ANN model and PWL-ANN algorithm. A decision support system (DSS) was implemented based on the EURECA algorithm. The DSS supports the users in tasks such as generating MLR rules using the EURECA algorithm, examining the significance of the generated rules and predictor attributes, as well as solving the back calculation and prediction problems.

Three case studies were present to demonstrate the application of the EURECA algorithm. The first case study used datasets related to predicting the pressure gradient of multi-phase flow oil pipelines to illustrate how knowledge could be extracted from the generated EURECA model. The second case study used the datasets collected from the CO2 capture process system at CETRI to demonstrate how the DSS was used for solving the back calculation problem. The results showed that the novice approach was able to solve up to five unknown predictor attributes with an error or average absolute deviation (AAD) of 11.39% and the expert approach was able to solve up to two unknown predictor attributes with an AAD of 14.12%. The third case study used the datasets related to the amine solvent in the amine-based CO2 capture process to demonstrate the applications of the EURECA algorithm on (i) knowledge extraction, and (ii) solving the back calculation scenario. The results showed that the generated EURECA model identified the concentration of methyldiethanolamine (MDEA) as the most significant parameter on the density of the mixed solvent. The back calculation results showed that almost 70% of the solved values have an AAD of less than 30% when four unknown was presented. Therefore, it can be concluded that the EURECA DSS was useful for solving regression problems related to engineering domains.
ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my academic supervisor, Dr. Christine Chan, for her patient supervision, valuable suggestions, and continuous financial support, which led to the accomplishment of this work.

I also wish to acknowledge the contributions of Dr. Raphael Idem and Dr. Paitoon Tontiwachwuthikul in sharing with me their insights and the datasets on the carbon dioxide capture plant of the Clean Energy Technology Research Institute. I also wish to thank Dr. Fatemeh Pouryousefi, Dr. Terradet Supap, Dr. San He, and Dr. Helei Liu for their insights related to clarifying knowledge on the carbon dioxide capture technology.

I am also grateful for the John Spencer Middleton and Jack Spencer Gordon Middleton Scholarships and the generous financial support from the Faculty of Graduate Studies and Research, University of Regina, the Government of Saskatchewan, and the Canada Research Program.

I would also like to thank all the staff in the Faculty of Engineering and Applied Science, and the Faculty of Graduate Studies and Research, for their help and support, and always answering my questions.

Finally, thank you to my family and friends who have encouraged and believed in me throughout this journey.
DEDICATION

To my grandfather, my parents, and my better half, Rigel

for all your love, encouragement, and inspiration.
# TABLE OF CONTENT

ABSTRACT........................................................................................................ ii

ACKNOWLEDGEMENTS................................................................................ iv

DEDICATION....................................................................................................... v

TABLE OF CONTENT......................................................................................... vi

LIST OF TABLES.................................................................................................. xiii

LIST OF FIGURES................................................................................................ xvii

ABBREVIATIONS................................................................................................ xxii

NOMENCLATURE................................................................................................ xxii

CHAPTER 1 INTRODUCTION.............................................................................. 1

1.1 Objectives and Scope of Work................................................................. 4

1.1.1 Objectives............................................................................................ 4

1.1.2 Scope of Work..................................................................................... 4

1.2 Outline of the Dissertation...................................................................... 7

1.3 Contributions of this study.................................................................. 12

CHAPTER 2 BACKGROUND AND LITERATURE REVIEW........................... 12

2.1 Artificial Neural Network....................................................................... 12

2.1.1 Architectures of Neural Network....................................................... 12

2.1.2 Learning Process of Neural Network Models................................. 18

2.2 Motivation of Neural Network Rule Extraction.................................. 20

2.3 Rule Extraction Algorithms................................................................... 21

2.3.1 Classification Scheme for Rule Extraction Algorithms.................. 22

2.3.2 Evaluation Criteria for Rule Extraction Algorithms....................... 25

2.3.3 Issues in Rule Extraction Algorithms............................................. 26
2.4 Decompositional Rule Extraction Algorithm

2.4.1 Towell & Shavlik (1993)

2.4.2 Setiono & Liu (1995)

2.4.3 Setiono & Liu (1997)

2.4.4 Gupta, Park, & Lam (1999)

2.4.5 Krishnan, Sivakumar, & Bhattacharya (1999)

2.4.6 Kim & Lee (2000)

2.4.7 Setiono & Leow (2000)

2.4.8 Setiono, Leow, & Zurada (2002)

2.4.9 Setiono & Thong (2004)

2.4.10 Odajima, Hayashi, Tianxia, & Setiono (2008)

2.4.11 Wang et al. (2011)

2.5 Pedagogical Rule Extraction Algorithm

2.5.1 Saito & Nakano (1988)

2.5.2 Thrun (1993)

2.5.3 Craven & Shavlik (1996)

2.5.4 Huysmans, Baesens & Vanthiene (2006)

2.5.5 Saad & Wunsch (2007)

2.5.6 de Fortuny & Martens (2012)

2.5.7 Augasta & Kathirvalavakumar (2012)

2.6 Eclectic Rule Extraction Algorithm

2.6.1 Hruschka & Ebecken (2006)

2.6.2 Khosrow & Sandesh (2006)
2.6.3 Kahramanli & Allahverdi (2009)..............................................45
2.6.4 Al Iqbal (2011).................................................................47
2.6.5 Mohamed (2011)...............................................................47
2.7 Comparisons on the Rule Extraction Algorithms.........................48
2.8 Multiple Linear Regression....................................................60

CHAPTER 3 PRELIMINARY METHODOLOGY – THE PWL-ANN ALGORITHM..63
3.1 Methodology...........................................................................65
3.2 Results and Analysis...............................................................71
  3.2.1 Overall Performance of the Three Approaches.......................79
  3.2.2 Performance of the Three Approaches on Each Individual Dataset…81
3.3 Two Hypotheses to Explain Poor Performance of the PWL-ANN
  Algorithm..............................................................................83
  3.3.1 Hypothesis 1.....................................................................83
  3.3.2 Hypothesis 2.....................................................................86
3.4 Discussion...............................................................................89
3.5 Conclusion.............................................................................90

CHAPTER 4 ENHANCED METHODOLOGY – THE ENHANCED-PWL-ANN
ALGORITHM................................................................................92
4.1 Methodology...........................................................................93
  4.1.1 Training the neural network for a given dataset.....................94
  4.1.2 Approximating the activation functions of the hidden neurons of the
       trained ANN model and generating the enhanced-PWL-ANN
       models.............................................................................95
4.1.2.1  Sampling algorithm………………………………………………101
4.1.2.2  Extended brute-force search algorithm…………………………101
4.1.2.3  Sensitivity analysis algorithm…………………………………106
4.1.3  Extracting rules from the generated Enhanced-PWL-ANN models………………………………………………………………………111

4.2.  Discussion: Comparison of PWL-ANN algorithm and enhanced-PWL-ANN algorithm………………………………………………………111
4.2.1  Comparison of techniques adopted in the algorithms………………111
4.2.2  Comparison of performances………………………………………..112
4.2.2.1  Comparing Accuracy and Fidelity of the Generated Models……………………………………………………………………………114
4.2.2.2  Comparing Comprehensibility of the Extracted Rules from the Generated Models……………………………………………………118

4.3.  “Opening the Black Box”: Extracting Rules from the Generated Models..122
4.4.  Conclusion………………………………………………………………130

CHAPTER 5 THE EURECA ALGORITHM…………………………………133
5.1  Heuristic and Motivation of the Proposed Algorithm…………………..133
5.2  The EURECA Algorithm………………………………………………135
5.2.1  Training the ANN model……………………………………………136
5.2.2  Clustering the dataset according to the hidden neuron weight values……………………………………………………………………………136
5.2.3  Perform MLR on each of the clustered group of data………..142
5.3  Accuracy of the EURECA models………………………………………142
7.4.1 Accuracy of the EURECA model.............................................. 194
7.4.2 Knowledge Extracted from the EURECA Model...................... 196
7.5 Discussion: Opening the “Black Box”............................................. 199
7.6 Conclusion.................................................................................... 204

CHAPTER 8 CASE STUDY #2 CARBON DIOXIDE CAPTURE SYSTEM – BACK
CALCULATION WITH EURECA MODEL................................................. 206

8.1 Background.................................................................................. 207
  8.1.1 Application Problem Domain...................................................... 207
  8.1.2 Previous studies on the CO₂ Capture Process System at CETRI..... 209
8.2 Preparation for Solving the Operational Problems......................... 214
  8.2.1 The Dataset.............................................................................. 214
  8.2.2 Training the ANN Models and Generating the EURECA Models... 216
8.3 Solving the Operational Problems using the Generated EURECA Models................................................................. 219
  8.3.1 Novice Approach...................................................................... 220
    8.3.1.1 Performance Evaluation of the Novice Approach............... 225
  8.3.2 Expert Approach..................................................................... 231
    8.3.2.1 Performance Evaluation of the Expert Approach.............. 242
8.4 Discussion..................................................................................... 245
  8.4.1 Comparison between the Novice and Expert Approach............. 245
  8.4.2 Comparison between the EURECA and ANFIS approach.......... 247
  8.4.3 Drawbacks of the EURECA Approach.................................... 249
8.5 Conclusion and Future Work.......................................................... 250
CHAPTER 9 CASE STUDY #3 UNDERSTANDING AND MONITORING AMINE
SOLVENT IN THE CO₂ CAPTURE SYSTEM................................................. 253

9.1 Background..................................................................................255
  9.1.1 Empirical Approach on Modeling Amine Solvents’ Properties...... 255
  9.1.2 Artificial Neural Network Approach........................................256

9.2 The Dataset.................................................................................. 257

9.3 Application of the EURECA Algorithm....................................... 258

9.4 Results and Analysis................................................................. 262
  9.4.1 Knowledge Extracted from the Generated EURECA Models ...... 262
    9.4.1.1 Detail Illustrate with the Density Dataset.............................263
    9.4.1.2 Verification with Experimental Observations...................... 272
  9.4.2 Monitoring the Amine Solvent using the Generated EURECA
      Models...................................................................................... 275

9.5 Conclusion..................................................................................285

CHAPTER 10 CONCLUSION AND FUTURE WORK............................... 287

10.1 Conclusions..............................................................................287

10.2 Future Work..............................................................................294

REFERENCES..................................................................................295

APPENDIX A – Rule Set A..............................................................319

APPENDIX B – Rule Set B..............................................................321

APPENDIX C – Justification and Evaluation of the Sampling Algorithm.... 323
  C.1 Data Size and Number of Breakpoints Vs Computation Time..........323
  C.2 Effect of Sampling on the Performance of the Approximation Algorithm.326
LIST OF TABLES

Table 2.1 – Comparison of rule extraction algorithms on classification and performance criterion……………………………………………………………………………50
Table 2.2 – Comparison of rule extraction algorithms on dataset used and rules comprehensibility…………………………………………………………….54
Table 3.1 – Characteristics of the datasets……………………………………………………………………………………………………………………………………………72
Table 3.2 – MSE given by the ANN model and the PWL-ANN approaches……………75
Table 3.3 – $R^2$ given by the ANN model and the PWL-ANN approaches………………76
Table 3.4 – MSE percentage difference between ANN and the three PWL-ANN approaches……………………………………………………………………….78
Table 3.5 – Performances of the PWL-ANN models……………………………………80
Table 3.6 – Dataset #6: Comparison of accuracies of the PWL approximation for each node with fidelity of the PWL-ANN model to the original ANN model……………..85
Table 3.7 – Data #15: Comparison of accuracies of the PWL approximation for each node with fidelity of the PWL-ANN model to the original ANN model……………..85
Table 3.8 – MSE’s given by PWL-ANN modeling for original dataset #6 and its subset………………………………………………………………………………88
Table 3.9 – Performance of PWL-ANN modeling on subset B of dataset #6 using approach 3……………………………………………………………………………88
Table 4.1 – Example of a sums of squared residuals (SSR) matrix: $M = 20$, $\text{maxBP} = 2$, $l = 3$…………………………………………………………………………………104
Table 4.2 – Example of Sensitivity Analysis Matrix…………………………………………………………..109
Table 4.3 – Characteristics of the datasets……………………………………………………………..113
Table 4.4 – Parameter settings of the PWL-ANN and the enhanced-PWL-ANN models………………………………………………………………………………113
Table 4.5 – MSE and R^2 given by Model #1, Model #2 and Model #3………………116
Table 4.6 – Comparison of 3 datasets: MSE percentage difference given by PWL-ANN models and Enhanced-PWL-ANN models…………………………………………………………………………………………………………………………116
Table 4.7 – Fidelity Vs Comprehensibility for Datasets #6, #7 and #15………………120
Table 4.8 – Comparing accuracies and comprehensibility of Rule Set A, Rule Set B and Rule Set C…………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………
Table 8.3 – Comparing predictive accuracies and number of rules given by different modeling approaches………………………………………………………………..... 218

Table 8.4 – Sample scenario #1 solved by the novice approach………………………… 221

Table 8.5 – The AAD between the calculated and target predicted attribute given by the test cases……………………………………………………………………………………….. 226

Table 8.6 – The AAD between the calculated and target predicted attribute given by the test cases……………………………………………………………………………………….. 229

Table 8.7 – Sample scenario #2 solved by the expert approach………………………… 235

Table 8.8 – Range of values covered by the matching equations with solutions for sample scenario #2………………………………………………………………………………….. 237

Table 8.9 – Results of solving the CO₂ production rate back calculation problems using the expert approach……………………………………………………………………………………….. 244

Table 9.1 – Accuracies of the trained ANN models and the generated EURECA models…………………………………………………………………………………………….. 260

Table 9.2 – Rules extracted by the EURECA algorithm (Rule Set A) for the Density dataset…………………………………………………………………………………………………………….. 265

Table 9.3 – Ranking of attributes’ significance given by Rule Set A and Rule Set B… 270

Table 9.4 – Observations made by Pouryousefi (2014) Vs the EURECA models………274

Table 9.5 – The AAD between the calculated and target predicted attribute given by the Density, Heat Capacity (1) and Refractive Index EURECA models…………………… 278

Table 9.6 – The AAD between the solved and actual predictor attributes given by the Density, Heat Capacity (1) and Refractive Index EURECA models…………………… 280
Table 9.7 – The AAD of individual components given by the RBFNN and BPNN from (Pouryousefi, 2014) and the EURECA Density model……………………………………. 284
Table C1 – Data size and number of breakpoints Vs computation time………………. 324
Table C2 – Characteristics of the datasets………………………………………………….. 329
Table C3 – Performance of approximation algorithm using different data size……. 329
Table C4 – Relative subsets size and relative performance of the approximation algorithm to the full dataset…………………………………………………………….331
LIST OF FIGURES

Figure 2.1 – A taxonomy of feed-forward and recurrent network architecture (Jain & Mao, 1996) ................................. 14

Figure 2.2 – An Artificial Neuron ......................................................................................................................... 16

Figure 2.3 – An Artificial Neural Network .................................................................................................................. 17

Figure 2.4 – The “translucency” criterion for classifying rule extraction algorithms (from Tickle, Andrews, Golea, & Diederich (1998)) .............................................................. 24

Figure 3.1 – Approximating the activation function (sigmoid function) with a three-piece linear approximation function .............................................................................................................. 68

Figure 3.2 – A trained neural network with three inputs, three hidden neurons and one output ........................................................................................................................................................ 70

Figure 3.3 – MSE percentage difference between the ANN models and the PWL-ANN models ........................................................................................................................................................................ 83

Figure 4.1 – Pseudo code of the enhanced-PWL-ANN algorithm ..................................................................................... 99

Figure 4.2 – Flow chart of the enhanced-PWL-ANN algorithm .......................................................................................... 100

Figure 4.3 – Classification tree obtained for dataset #6 using the rules from the generated enhanced-PWL-ANN model as predicted attributes ........................................................................ 125

Figure 4.4 – Regression tree obtained for dataset #6 ........................................................................................................ 127

Figure 5.1 – Hidden neuron, j ........................................................................................................................................... 137

Figure 5.2 – Clustering by weighted input values – dataset is clustered into three groups by the weighted input values in each hidden neuron .............................................................................. 140

Figure 5.3 – Data clusters are defined by the combination of groups a dataset belongs to in each hidden neuron ........................................................................................................................................ 141
Figure 6.1 – Schematic diagram of the EURECA algorithm and the EURECA decision support system……………………………………………………………………… 152
Figure 6.2 – Structural framework of the EURECA DSS………………………….. 152
Figure 6.3 – Main components of the EURECA DSS Modules…………………….. 160
Figure 6.4 – Hidden weight values in matrix format required by Data I/O Module….. 163
Figure 6.5 – Summary of each individual rule given by the Rules and Data Evaluation Module……………………………………………………………………… 166
Figure 6.6 – Both real and normalized values of the dataset uploaded to the EURECA DSS system via the Data I/O module…………………………………….. 172
Figure 6.7 – Plot of predicted output values given by the generated EURECA model Vs Actual output values given by the EURECA modeling module……………….. 173
Figure 6.8 – Accuracy, significance and attribute ranking of the generated EURECA model given by the Rules and Data Evaluation Module…………………………………….. 176
Figure 6.9 – Summary of Rule #1 given by the Rule and Data Evaluation Module….. 177
Figure 6.10 – Summary of Rule #2 given by the Rule and Data Evaluation Module…. 178
Figure 6.11 – New input data uploaded for prediction in the prediction module of the EURECA DSS……………………………………………………………………………………………………………………… 180
Figure 7.1 – 3D plot of pressure gradient Vs oil superficial velocity Vs water superficial velocity………………………………………………………………………………………………………………… 187
Figure 7.2 – First 10 data tuples clustered according to their weighted input values……………………………………………………………………………………………………………………… 190
Figure 7.3 – Summary of Rule #1 given by the Im() function in R…………………. 192
Figure 7.4a – Curve given by ANN model for data tuples with Pd = 56.3 mm……….. 201
Figure 9.2 – AAD between the calculated and target predicted attribute Vs number of unknown variables ................................................................. 278

Figure 9.3 – AAD between the solved and actual predictor attributes Vs number of unknown variables ................................................................. 280

Figure 9.4 – AAD percentage error distribution of the solved unknown attributes given by the density ERUECA model ................................................................. 281

Figure A1 – Rules extracted from dataset #6 (Rule Set A) ........................................... 319

Figure A2 – Range of values (Normalized) covered by each attribute for the Rule Set A ................................................................. 320

Figure B1 – Rule Set B Equations ................................................................. 322

Figure C1 – Computation time Vs data size for 1 breakpoint ........................................... 325

Figure C2 – Computation time Vs data size for 2 breakpoints ........................................... 325
### ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAD</td>
<td>Average Absolute Deviation</td>
</tr>
<tr>
<td>ANFIS</td>
<td>Adaptive-Network-based Fuzzy Inference System</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial Neural Network</td>
</tr>
<tr>
<td>BP</td>
<td>Breakpoint</td>
</tr>
<tr>
<td>BPNN</td>
<td>Backpropagation Neural Network</td>
</tr>
<tr>
<td>CETRI</td>
<td>Clean Energy Technology Research Institute</td>
</tr>
<tr>
<td>CO₂</td>
<td>Carbon Dioxide</td>
</tr>
<tr>
<td>DSS</td>
<td>Decision Support System</td>
</tr>
<tr>
<td>EURECA</td>
<td>Extracting Understandable Rules by Employing Clustering with Artificial Neural Network</td>
</tr>
<tr>
<td>ITC</td>
<td>International Test Center for CO₂ Capture</td>
</tr>
<tr>
<td>MDEA</td>
<td>Methyl diethanolamine</td>
</tr>
<tr>
<td>MEA</td>
<td>Ethanolamine</td>
</tr>
<tr>
<td>MLP</td>
<td>Multilayer Perceptron</td>
</tr>
<tr>
<td>MLR</td>
<td>Multiple Linear Regression</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean Square Error</td>
</tr>
<tr>
<td>NOₓ</td>
<td>Nitrogen Oxide</td>
</tr>
<tr>
<td>PWL</td>
<td>Piecewise Linear</td>
</tr>
<tr>
<td>PWL-ANN</td>
<td>Piecewise Linear Neural Network</td>
</tr>
<tr>
<td>RBFNN</td>
<td>Radial Basis Function Neural Network</td>
</tr>
<tr>
<td>SA</td>
<td>Sensitivity Analysis</td>
</tr>
<tr>
<td>SOₓ</td>
<td>Sulfur Oxide</td>
</tr>
<tr>
<td>SSR</td>
<td>Sums of Squared Residuals</td>
</tr>
</tbody>
</table>
### NOMENCLATURE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\xi_j$</td>
<td>Weighted input of node $j$ in an ANN model</td>
</tr>
<tr>
<td>$\xi_0, \xi_0$</td>
<td>Locations of breakpoints</td>
</tr>
<tr>
<td>$x_i$</td>
<td>Value of input $i$</td>
</tr>
<tr>
<td>$y_i$</td>
<td>Value of output $i$</td>
</tr>
<tr>
<td>$y_k$</td>
<td>Value of output $k$</td>
</tr>
<tr>
<td>$\hat{y}_i$</td>
<td>Predicted value of output $i$ given by the ANN model</td>
</tr>
<tr>
<td>$\tilde{y}_i$</td>
<td>Predicted value of output $i$ given by the rule extraction model</td>
</tr>
<tr>
<td>$\bar{y}$</td>
<td>Average value of output $y$</td>
</tr>
<tr>
<td>$w_{ij}$</td>
<td>Weight of input $i$ to hidden node $j$ in an ANN model</td>
</tr>
<tr>
<td>$w_{jk}$</td>
<td>Weight of hidden node $j$ to output neuron $k$ in an ANN model</td>
</tr>
<tr>
<td>$\theta_j$</td>
<td>Bias value of node $j$ in an ANN model</td>
</tr>
<tr>
<td>$f_j(\xi)$</td>
<td>Activation function of hidden node $j$ in an ANN model</td>
</tr>
<tr>
<td>$f_k(\xi)$</td>
<td>Activation function of output node $k$ in an ANN model</td>
</tr>
<tr>
<td>$L(\xi)$</td>
<td>Piece linear function approximating the sigmoid function</td>
</tr>
<tr>
<td>$O_j$</td>
<td>Output of the hidden $j^{th}$ neuron</td>
</tr>
<tr>
<td>maxBP</td>
<td>Maximum breakpoint defined by the user in the enhanced-PWL-ANN algorithm</td>
</tr>
<tr>
<td>$l$</td>
<td>Minimum distance between breakpoints in the enhanced-PWL-ANN algorithm</td>
</tr>
<tr>
<td>$E$</td>
<td>Error in terms of MSE percentage difference defined by the user in the enhanced-PWL-ANN algorithm</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>$D_{\text{sample}}$</td>
<td>Size of sample dataset defined by the user in the enhanced-PWL-ANN algorithm</td>
</tr>
<tr>
<td>$E_{x0}$</td>
<td>Average absolute deviation between calculated and target predictor variable</td>
</tr>
<tr>
<td>$E_{y0}$</td>
<td>Average absolute deviation between calculated and target predicted variable</td>
</tr>
<tr>
<td>$T_{\text{equ}}$</td>
<td>Total number of matching equations found in a back calculation problem</td>
</tr>
<tr>
<td>$T_{\text{equ}, 5%}$</td>
<td>Total number of matching equations found that provides a solution within $Y/+/- 5%$</td>
</tr>
</tbody>
</table>
CHAPTER 1 INTRODUCTION

Artificial neural network (ANN) is a popular data mining approach best known for its high predictive accuracy and resistance to noise. However, this algorithm is considered as a “black box” algorithm since it does not provide explicit relationships between the predictor (or input) and predicted (or output) attributes. In order to overcome the deficiency of lack of explanation, researchers have developed ANN rule extraction algorithms to generate language syntax from trained ANN models. However, most of these algorithms focus on classification problems, when in reality a lot of the problems encountered are regression problems. Therefore, the main objective of this study is to develop an ANN rule extraction algorithm for non-linear regression problems.

The artificial neural network or ANN algorithm was applied to various problem domains for predictive modeling and had successfully generated highly accurate predictive models. For example, in engineering, the ANN approach was applied to detect faults in power transmission system (Jamil et al., 2015), predict air pollution level (Elangasinghe et al., 2014), monitor the capture of carbon dioxide (CO\textsubscript{2}) (Zhou et al., 2011; Sipöcz et al., 2011), predict clinical outputs, and diagnose disease in biomedical engineering (Cai & Jiang, 2015). In business, ANN was used to help prevent financial distress and bankruptcy, monitor corporate securities, provide business decision support, and predict credit scores (Tkáč & Verner, 2016). In medicine, researchers have utilized ANN to assist with disease diagnosis such as hepatitis and tuberculosis, suggest treatment regimens, and differentiate between different types of cancers (Sheikhtaheri et al., 2014). However, despite their high predictive accuracies, the ANN models are black boxes that do not
provide researchers with explicit relationships between the predictor and predicted variables, often needed for gaining a better understanding of the studied problem (Chan, 2007). ANN models consist of complex non-linear functions, which are not interpretable. To address this deficiency, researchers proposed the rule extraction approach, which can generate explicit information based on the ANN models.

The rule extraction algorithms are classified into three approaches based on the translucency of the algorithm. The translucency of an approach refers to how exposed the underlying or originally trained ANN model is in the rule extraction algorithm. The completely translucent decomposition approach usually involves more complex computation and extracts rules by examining the activation functions and weights of the hidden neurons. On the other end of the translucency spectrum is the opaque pedagogical approach, which views the underlying ANN model as a “black-box” and extracts rules by studying the relationships between the predictor and predicted values given by the underlying ANN model. In between the two ends of the translucency spectrum is the eclectic approach, which usually takes on a hybrid approach by incorporating other machine learning algorithms to extract rules from the trained ANN models.

ANN rule extraction algorithms were applied for modeling datasets in various disciplines with good results, and some sample studies are presented as follows. Chen et al. (2010) applied ANN rule extraction to help construction companies minimize financial risks. Chan & Jian (2013) used the ANN rule extraction algorithm and predicted the level of air pollution and identified the key parameters that contributed the most to the degree of air.

Despite the large number of rule extraction algorithms proposed in the literature, only a few of these address the regression problem (Augasta & Kathirvalavakumar, 2012). Regression problems refer to problems where the outputs are in continuous numeric values, as oppose to classification problems where output variables are in the form of class labels. In statistics, regression analysis refers to the process of determining the relationship between one dependent variable, $y$, and a series of independent variable, $x_i$. If there is only one independent variable, $x$, then the problem is known as linear regression, and the relationship between $x$ and $y$ is in the form of

$$ y = ax + b $$

where $y$ is the predicted variable, $a$ is the coefficient of the predictor variable $x$ and $b$ is a constant, which is the error that represents the difference between the predicted and observed value of $y$.

If there are more than one independent variables, then the problem is known as multiple regression and the relationship between $x_i$ and $y$ will be in the form of

$$ y = a_1 x_1 + a_2 x_2 + ... + a_n x_n + b $$
where y is the predicted variable, \( a_1 \) is the coefficient of the predictor variable \( x_1 \), \( a_2 \) is the coefficient of the predictor variable \( x_2 \), \( a_n \) is the coefficient of the predictor variable \( x_n \), and b is a constant variable that represents the difference between the predicted and observed value of y.

Therefore, this study proposes three rule extraction algorithms to address regression problems.

1.1 Objectives and Scope of Work

1.1.1 Objectives

The main objectives of this research study are:

1. To develop an artificial neural network rule extraction algorithm for non-linear regression problems; the algorithm extracts interpretable equations from the trained ANN models so as to provide insights about the relationships between the predictor and predicted attributes of a studied problem;

2. To open the “black-box” of the ANN model by investigating the relationships among the data tuples that are assigned to a similar range of weighted input values to reveal the learning process of the ANN;

3. To demonstrate the applicability of the developed rule extraction algorithm for solving industrial engineering problems;

4. To develop a decision support system (DSS) based on the rule extraction algorithm so as to: (i) support providing insights about the studied problem to decision makers, and (ii) assist in solving the back calculation problem, which
requires determining the values of the unknown predictor attributes so that the target predicted values could be achieved.

1.1.2 Scope of Work and Assumptions

The scope of work of this study is as follows:

1. Three ANN rule extraction algorithms were developed for non-linear regression problems.

2. A decision support system was developed based on the ANN rule extraction algorithms with the best performance.

3. Three industrial datasets were used to demonstrate the best algorithm’s capability on solving industrial problems.

4. The performance of the generated model given by the best algorithm was evaluated at two levels:

   i. At the model level, the accuracy, fidelity, and comprehensibility of the generated models were evaluated. Accuracies were measured by the value of $R^2$ and mean square error (MSE), as these are some common indicators used to evaluate accuracies of machine learning models, and the least square approach identifies the multiple linear regression models by minimizing the mean square error (Wang et al., 2011, Zhou et al., 2011a). The fidelities of the generated models to the originally trained ANN model were measured by the MSE percentage difference between the generated model and the originally trained ANN model. Comprehensibilities of the generated models were
measured by the number of rules extracted, and the number of conditions in
the rules.

ii. At the rule or equation level, the significance of each rule was evaluated using
the statistical measures in the ANOVA table. The significance of each
predictor attributes was evaluated by the p value.

Some assumptions of this study are as follows:

1. The underlying neural network

The architecture of the underlying ANN model is limited to a three-layer feedforward
backpropagation neural network with a sigmoid or tangent activation function because
these are the most common types of neural network model, and based on experience, one
hidden layer is sufficient to solve most non-linear problems without overfitting.

2. Targeted problems: Regression

As mentioned earlier, there is a lack of algorithms for nonlinear regression problems.
Regression problems refer to problems where the output variables are described by
continuous values and not classes. Most of the reviewed algorithms were designed for
classification problems. Among these, and many that claim to be applicable to regression
or classification problems with continuous values in fact required the values to be
discretized. To improve the applicability of neural network rule extraction algorithms on
real-world applications, it is important to consider regression problems. Thus, the
proposed algorithms would be designed to target nonlinear regression problems.

3. Result representation: Numeric Function

In the traditional linear regression approach, the output is often given in the form of \( Y = f(X) \), which provides a good explanation on how the inputs and outputs are correlated.
Hence, for ease of interpretation, the generated output from the proposed algorithm is represented in the form of “rules” that are in the form of linear numeric functions.

4. Datasets used in this study involve Engineering data and UCI data

The UCI datasets are used to validate the performances of the algorithms in terms of accuracy, fidelity, and comprehensibility of the proposed algorithm. However, while the UCI datasets are based on “real-world” problems, some of them are quite simple and cannot justify the applicability of the proposed rule extraction algorithms on real world complex problems. Therefore, real industrial engineering datasets are used to demonstrate the applicability of the proposed algorithms on complex engineering problems.

Some areas that are out of the scope of work of this study include: feedforward neural network with more than one hidden layer, ANN with activation function other than the linear function in the output layer, and classification problems.

1.2 Outline of the Dissertation

There are ten chapters in this dissertation. Chapter 1 introduces the research topic and the objectives, the scope of work, and some assumptions of this research. Chapter 2 presents some background on artificial neural network, the motivation behind ANN rule extraction algorithm, a literature review on the existing ANN rule extraction algorithms, and relevant work on multiple linear regression. Chapter 3 presents the first proposed algorithm called the piecewise linear ANN rule extraction algorithm, abbreviated to PWL-ANN. The performance of the PWL-ANN model was evaluated using 15 UCI datasets and 4 real world engineering dataset. Based on an analysis of the results of the
preliminary algorithm, an enhanced version called enhanced-PWL-ANN, was developed and the algorithm was presented in Chapter 4. The work on the PWL-ANN algorithm was already published in (Chan & Chan, 2017a and 2017b). The enhanced-PWL-ANN algorithm improved upon the accuracy and fidelity of the preliminary-PWL-ANN algorithm. By applying the decision tree algorithm to the generated rule set, the less significant rules were removed and comprehensibility of the rule set was improved. However, the models generated by the enhanced-PWL-ANN algorithm suffer from two major drawbacks. First, the rules were generated in terms of normalized values, which may have no physical mean to the domain expert; an additional step is required in order to translate the equations to ones using the original values. Secondly, the accuracies of the generated rules were not high enough to solve the back calculation problem accurately. To overcome these weaknesses, Chapter 5 presents an eclectic ANN rule extraction algorithm, called EURECA (which stands for “Extracting Understandable Rules by Employing Clustering with Artificial Neural Network”). The generated EURECA models have higher predictive accuracies than both the originally trained ANN models and the enhanced-PWL-ANN models, and the EURECA algorithm generates rules with the real values. Chapter 6 presents the development of the EURECA decision support system. The DSS generates rules based on the EURECA algorithm and provides the user with insights on the relationships between the predictor and predicted attributes of the studied problem. The DSS also supports the user in performing back calculation in a problem-solving scenario, which requires determining the values of the unknown predictor attributes so that the target predicted values could be achieved. A multi-phase flow pipeline dataset obtained from a subsea pipeline of Bohai Oilfield, China, was used
to demonstrate the application of the EURECA DSS. Chapter 7 presents the first case study using a second dataset from the application domain of multi-phase flow pipeline obtained from experimental studies. The focus of this case study was to show how the ANN “black-box” can be opened using the EURECA model. Chapter 8 presents the second case study using the dataset collected from the amine-based CO₂ capture pilot plant at the Clean Energy Technology Research Institute (CETRI) (formerly known as the International Test Center for CO₂ capture (ITC)) located at the University of Regina, Regina, Saskatchewan, Canada. The focus of this case study is to demonstrate how the generated EURECA models can be used to solve operational problems which involved the determination of some predictor attribute values (e.g. steam flow rate through reboiler, amine circulation rate) such that the target predicted attribute values (e.g. CO₂ production rate, CO₂ absorption efficiency) can be achieved. In this study, this specific kind of problem scenario is called the back calculation problem and two approaches were proposed to solve this back calculation problem. Chapter 9 presents the third case study using an amine solvent dataset obtained through experiments. This case study demonstrates the application of the EURECA algorithm, which demonstrates how the algorithm can both provide explicit relationships between the predictor and predicted variables, and solve the back calculation problem. Last but not least, Chapter 10 concludes the research and provides some recommendations for future work.

1.3 Contributions of this study

The major contributions of this study can be summarized below:
1. The proposed ANN rule extraction algorithms, namely the PWL-ANN, enhanced PWL-ANN, and EURECA algorithm, generate explicit relationships between the predictor and predicted variables in the form of multiple linear regression equations so that a better understanding of the studied problem can be gained. This opens up the “black-box” of the ANN model. By examining the correlation between the assigned weight values and the values of the predictor attributes, an understanding about the learning process of the ANN algorithm is obtained. It was discovered that there was a correlation between the assigned weights of the input data tuples and the hypothesized linear relationship that exists among subsets of data in a given problem domain.

2. The EURECA algorithm proposed adopt a novel approach of clustering the data tuples according to their weights to the hidden neurons.

3. The three case studies demonstrated that the developed EURECA algorithm was suitable for modeling real industrial engineering problems, and by applying the EURECA algorithm, valid and comprehensible rules could be generated to enhance the domain experts’ understanding of the studied problem. The EURECA algorithm can also be used for solving the back calculation problems.

4. A decision support system (DSS) based on the EURECA algorithm was implemented. This is a major contribution because DSS that are based on ANN rule extraction algorithm is rare. The DSS supports the users in terms of: (i) generating multiple linear regression rules for a studied problem using the EURECA algorithm, (ii) providing some insight to the user on the relationships between the predictor and predicted attributes of the studied problem, (iii)
performing back calculation, which requires determining the values of some unknown predictor attributes so that the target predicted values could be achieved, and (iv) generating accurate predictions given a new set of predictor attributes of the studied problem.

5. Most of the studies on ANN rule extraction algorithms found evaluated their algorithms by applying them onto UCI datasets, whereas the case studies in this research showed that the EURECA algorithm is suitable for solving industrial engineering problems.

The details of these contributions are elaborated in the following chapters.
CHAPTER 2 BACKGROUND AND LITERATURE REVIEW

2.1 Artificial Neural Network

The concept of neural network was first proposed in 1943 during the second world war by McCulloch and Pitts. Their idea was to create mind-like machines by interconnecting models that mimics the behavior of biological neurons. It was a very simple model and was not capable of “learning”. In 1958, Rosenblatt introduced a learning machine called “perceptron” which is the predecessor of today’s neural network models (Meireles, Almeida, & Simões, 2003). However, it was not until the 1980s, when computers had gained sufficient processing power that the research on artificial neural network began to take off. Many neural network models were developed since then, such as the multilayer perceptron (MLP) or multilayer feedforward neural network, recurrent network models, radial basis function (RBF) model etc. The applications of neural networks include pattern recognition, image processing, optimization, modeling and control, image processing, classification etc.

2.1.1 Architectures of Neural Network

An artificial neural network model is a group of neurons that are interconnected. It usually consists of an input layer of neurons, one or more layers of hidden neurons, followed by an output layer. As described by Jain & Mao (1996), “ANNs model can be viewed as weighted directed graphs in which artificial neurons are nodes and directed edges (with weights) are connections between neuron outputs and neuron inputs.” Depending on the connection pattern, most ANN models can be classified as feedforward networks or recurrent (or feedback) networks, as shown in Figure 2.1. As suggested by the names, feedforward networks only has unidirectional edges that goes “forward”,

12
whereas for recurrent networks, loops can be formed by feedback connections. Since the most widely applied neural network is the multilayer perceptron (Meireles et al., 2003), which belongs to the feedforward network family, we shall focus on feedforward neural networks and the training for such networks.
Figure 2.1 – A taxonomy of feed-forward and recurrent network architecture (Jain & Mao, 1996)
Let’s consider a three-layer feedforward neural network. Suppose there are $i$ number of inputs $(x_1, x_2, \ldots, x_i)$, then the $j^{th}$ neuron in the hidden layer will receive $i+1$ number of inputs with signals $x_i$ through $x_i$, each associated with a weight ($w_i$, $\ldots$, $w_i$), and a bias value, $\theta$. The neuron also has a transfer function or activation function, $f_j$, as shown in Figure 2.2.

The neuron has two functions: first, it sums up the product of each input and its weight and the bias. The bias is a constant value that allows the shifting of the activation function in order to give better fit to the data. The neuron then passes the summed value through a transfer function (or activation function) and transforms the sum to some finite value, usually in the range of [0, 1] or [-1, 1], depending on the chosen function. The transfer function introduces a nonlinearity into the network and output of the $j^{th}$ neuron is, $O_j$:

$$ O_j = f_j\left( \sum_{i=1}^{i} w_{ij}x_i + \theta_j \right) \quad (2.1) $$

Then, the artificial neural network is an interconnection of these individual neurons, as shown in Figure 2.3. The $k^{th}$ output of the neural network will be given by:

$$ y_k = f_k\left( \sum_{j=1}^{j} w_{jk}O_j + \theta_k \right) \quad (2.2) $$

Some choices of transfer function of a neuron include a threshold function, piecewise linear function, sigmoid function, radial basis function and Gaussian function. A linear function is often used in the output layer. The design of a neural network model is out of the scope of this study and will not be further discussed.
Figure 2.2 – An Artificial Neuron
Figure 2.3 – An Artificial Neural Network
2.1.2 Learning Process of Neural Network Models

The artificial neural network learns by iteratively updating the connection weights within the network until performance is satisfactory. The learning of a neural network model involves the learning paradigm, which refers to what information is available to the model, and the learning algorithm, which refers to how the weights are adjusted in the learning process.

The learning paradigms of the learning process can usually be classified as supervised or unsupervised. During a supervised training, the network is provided with the correct output of each of the input sets. Weights are adjusted accordingly by comparing the predicted outputs to the given outputs. During an unsupervised training, no correct outputs are given. This type of training is usually used when ANN is being trained to group inputs into several classification groups. There is also hybrid learning, which is a combination of supervised and unsupervised training. This study will focus on the Backpropagation algorithm, which is a type of supervised learning paradigm, because it is the most popular learning algorithm used in feedforward neural network models.

Different learning algorithms have different learning rules that they use to adjust the connection weights. For supervised training, the network “learns” through error-correction rules which use the error signal, $E$, to adjust the connection weights through each iteration, until the error is reduced to minimum. The squared error cost function is
the most frequently used error correction rule for neural network ((Jain & Mao, 1996), and it is defined as follows:

\[ E = \frac{1}{2} \sum_{k \in K} (O_k - t_k)^2 \]  

(2.3)

where \( t_k \) = predicted output by the neural network.

In a backpropagation algorithm, the rate of change of error with respect to the given connection rate is considered. We will not get into the detail calculation of the algorithm, since we assume our rule extraction algorithm will be applied to a trained neural network.

The steps involved in the backpropagation algorithm are as follow:

1. Initialize the weights of the neural network model.
2. Pass a set of input data to the model to obtain an output.
3. For each output node, calculate the error signal, \( \delta_k \), by taking the partial derivative of the output function with respect to the weight:

\[ \delta_k = (y_k - t_k) \frac{\partial f_k(O_j)}{\partial w_{jk}} \]  

(2.4)

where \( y_k \) is the \( k^{th} \) output neuron,

\( f_k \) is the activation function of the \( k^{th} \) output neuron, and

\( w_{jk} \) is the weight between the \( j^{th} \) hidden neuron and the \( k^{th} \) output neuron.

4. For each of the hidden neuron, calculate the error signal, \( \delta_j \)

\[ \delta_j = \frac{\partial f_j(x_i)}{\partial w_{ij}} \sum_{k \in K} \delta_k w_{jk} \]  

(2.5)

where \( x_i \) is the input from the \( i^{th} \) input neuron,

\( f_j \) is the activation function of the \( j^{th} \) hidden neuron, and

\( w_{ij} \) is the weight between the \( i^{th} \) input neuron and the \( j^{th} \) hidden neuron.
5. The weight and bias is adjusted using:

\[ w = -\eta \delta_l \theta_{l-1} \]  
\[ \theta = -\eta \delta_l \]  

where \( l \) is the \( l^{th} \) layer in the neural network and \( \eta \) is a small constant.

6. Repeat from step 2, until the error is below a pre-specified threshold or maximum number of iterations is exceeded.

2.2 Motivation of Neural Network Rule Extraction

The first and most obvious motive of rule extraction is to enhance the explanation power of neural network so that the users can gain explicit understanding on how the input and output attributes are related. However, since “white-box” models, such as decision trees, could provide such understanding, why is it necessary to extract rules from a “black-box” model? As pointed out by (Huysmans et.al, 2006) there are several advantages of using such an intermediate opaque model: (1) a well trained opaque model can usually represent the dataset better than the dataset itself, because it can filter out noise present in the dataset; (2) the black-box model can be used to generate a larger dataset; (3) in many applications, such opaque models as ANNs or SVMs achieves better performance in terms of accuracies than “white-box” model. The facts that (i) a well trained opaque model could better represent the dataset and (ii) opaque models such as ANN usually achieve better performance, give us the second motivation of rule extraction from NN
models: to develop an accurate rule induction algorithm using ANN as an intermediate tool.

Zhou (2004) made the interesting observation that the two motives of enhancing the explanatory power of a neural network and developing an accurate rule induction algorithm using ANN as an opaque model can be conflicting. The first motive concerns the fidelity of the rule extraction algorithm to the underlying model. The second motive concerns the accuracy of the rule extraction algorithm. If the assumption is that the underlying ANN models gives good accuracies, then if the proposed rule extraction algorithm has a high fidelity to the ANN model, the rule extraction algorithm will also demonstrate good accuracy. This will fulfill both of the objectives on fidelity and accuracy. However, if there is a dataset that the underlying ANN model does not predict well, then the rule extraction algorithm could only accomplish one of the two objectives: either it models the ANN model very well, or it generates a set of rules that gives good accuracy. In this study, we assume that the underlying ANN model is already giving satisfactory performance, thus, our focus will be on the fidelity of the rule extraction algorithm in modeling the ANN model.

2.3 Rule Extraction Algorithms

In this section, the classification and evaluation criteria of ANN rule extraction algorithms, as well as some current issues of ANN rule extraction will be discussed. These criteria will be used later in Chapter 3 when the existing ANN rule extraction algorithms are compared.
2.3.1 Classification Scheme for Rule Extraction Algorithms

Andrews, Diederich, & Tickle (1995) suggested a classification scheme for rule extraction algorithms that has been widely used by researchers in ANN rule extraction algorithms and even support vector machines (SVMs) rule extraction algorithms (Martens, Setiono, Vanthienen, & Baesens, 2008). The classification scheme includes the following five criteria to classify the type of rule extraction algorithms:

1. The translucency of the rule extraction algorithm regarding the underlying neural network. Translucency refers to how much the algorithm dissects the underlying ANN model. This criterion classifies the algorithms into three categories: decompositional, pedagogical and eclectic, as shown in Figure 2.4. On the most left side of the spectrum are the decompositional algorithms. The algorithms extract rules by examining activation and weights of the hidden layer neurons and the algorithms are considered to be completely translucent. On the other end are the pedagogical algorithms, which extract rules by mapping the relationships between the inputs and outputs as closely as possible to those given by the trained ANN models without exploring the ANN models. The underlying ANN models are still treated as a “black-box” and “translucency” is not considered at all. The eclectic algorithms are hybrid of the two and lies in between the two classifications on the translucency spectrum.

2. The expressive power of the extracted rules.

Usually rules can be expressed in the following forms:

- Propositional rules: These rules have the If… Then.. expression based on conventional propositional logic;
• M-of-N rules: These rules have a at least/exactly/at most condition and are usually expressed as follows (Martens et al., 2008):

If \{at least/exactly/at most\} M of the N conditions (C1, C2,\ldots, CN) are satisfied, Then Class = 1.

• Decision trees

• Numerical functions: This is more for regression problems where the outputs are continuous numerical values than classification problems where the outputs are classes.

3. The quality of the rules, including rule accuracy, rule fidelity and rule comprehensibility, these qualities will be further discussed in the algorithm evaluation criteria section in Ch 2.3.2.

4. The algorithm complexity refers to how complex is the rule extraction algorithm.

5. Requirement for specialized training regime, such as the requirement of a steep sigmoid function as the hidden activation functions, or restrictions on the network architecture, e.g. applicable to feedforward ANN only.

Most of the area concerning the last three classification criteria suggested by Andrews et al. (1995) overlap with the evaluation criteria suggested by Craven & Shavlik (1999), and the details will be given in the next section.
Figure 2.4 – The “translucency” criterion for classifying rule extraction algorithms

(from Tickle, Andrews, Golea, & Diederich (1998))
2.3.2 Evaluation Criteria for Rule Extraction Algorithms

Craven & Shavlik (1999) suggested five criteria in order to evaluate performance of a given rule extraction algorithm:

1. Comprehensibility: how comprehensible are the extracted rules?

In this study, the comprehensibility will be evaluated by the number of rules (or nodes, if the outcome is a decision tree) given by the rule extraction algorithm, as well as the number of conditions given by each rule.

2. Fidelity: how accurately are the rules modeling the neural network model?

For classification problems, fidelity can be given by:

\[
Fidelity = \frac{\text{Accuracy of Extracted Rules}}{\text{Accuracy of ANN}} \times 100\% \tag{2.8}
\]

For regression problems, fidelity can be given by:

Assume there are \( i \) datasets,

\[
Fidelity = \frac{\sum_i (\hat{y}_i - \bar{y}^2)}{\sum_i (\bar{y}^2 - y_i)} \times 100\% \tag{2.9}
\]

where \( \hat{y} = \text{predicted value by ANN} \)

\( \bar{y} = \text{predicted value by extracted rules} \)

\( \bar{y} = \text{average value of the output} \)

\[
\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i
\]

\( y_i = \text{the expected output of the } i^{th} \text{ dataset.} \)

3. Accuracy: how accurate are the predictions or classifications given by the rules?

For classification problems, accuracy can be given by
For regression problems, accuracy can be given by $R^2$ or mean squared error (MSE) or root mean squared error (RMSE), those values can be given by:

$$R^2 = 1 \ - \ \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$  \hspace{1cm} (2.11)$$

where $\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$ and $\hat{y} =$ predicted output

$$MSE = \frac{\sum_i (y_i - \hat{y}_i)^2}{n}$$  \hspace{1cm} (2.12)$$

$$RMSE = \sqrt{MSE}$$  \hspace{1cm} (2.13)$$

4. Scalability: can the rule extraction algorithm be applied to networks with large input space and large number of units and weighted connections? Scalability is difficult to measure and is omitted in most of the research study.

5. Generality: does the algorithm require special training regimes, e.g. the activation function has a steep sigmoid function, or restrictions on the network architecture, e.g. feedforward ANN only?

### 2.3.3 Issues in Rule Extraction Algorithms

Craven & Shavlik (1999) suggested that, in order to make an “impact” and increase the application of the rule extraction algorithms, researchers need to focus on solving the problems of scalability and generality, as well as improving the availability of the developed algorithms.
The issue of scalability concerns both running time and comprehensibility of the extracted rules. Decompositional algorithms are usually computationally complex since they usually examine all the hidden nodes in the ANN model and the number of generated rules is proportional to the network size. Pedagogical algorithms are usually not as computationally complex, however, they will still generate a large rule set if the problem domain has a large number of features. Two solutions suggested by Craven & Shavlik (1999) to tackle the issue of scalability are:

1. Improving the comprehensibility by compromising the algorithm’s fidelity to the neural network.

2. Research on algorithms that can extract rules at anytime. By “anytime”, the authors meant an algorithm that could be interrupted at any point during their computation and an extracted rule set would still be given.

In the comparisons of rule extraction algorithms in the next chapter, we would be able to see that the concept of the first suggestion was implemented in some algorithms, however, there is hardly any research that implement the concept of the second suggestion.

Regarding generality, Craven & Shavlik (1999) argued that “in order to have a large impact, rule-extraction methods will have to exhibit a high level of generality”. In this aspect, pedagogical approaches are usually better than decompositional approaches. Since the neural network is treated as a “black box” in pedagogical approaches, they can generally be applied to any neural network topologies, and even other “black box” algorithm such as support vector machines (SVMs). However, generality also concerns
the format of the attributes, e.g. some algorithms require input/output encoding and that decreases the generality of the algorithm as well.

The last improvement that Craven & Shavlik (1999) suggested to researchers is to make the developed algorithms more readily available by providing a portable source code or set them up as on-line servers.

2.4  Decompositional Rule Extraction Algorithm

2.4.1  Towell & Shavlik (1993)

Towell & Shavlik (1993) proposed one of the earliest decompositional neural network rule extraction algorithm called the M-of-N method. The algorithm involves the following steps: (1) for each hidden and output unit, weighted links with the similar values are clustered into groups; (2) All the line weights of the member in the same group will be set to the average weights of that group; (3) Groups that have no significant affect on the units will be removed; (4) Biases can be optimized by freezing the weights and adjusting the biases using backpropagation; (5) rules in the form of M-of-N can be extracted for each hidden and output unit; (6) simplify rules where possible. The authors examined the accuracies, fidelities, and rule comprehensibility of the algorithm, by applying it to two datasets related to molecular biology. Results were compared to the trained NN, SUBSET (Fu, 1991), EITHER (Ourston, 1991), C4.5 (Quinlan, 1993) and LINUS (Džeroski and Lavrač, 1991). The results showed that M-Of-N made significantly less error than SUBSET, EITHER, LINUS and C4.5 in one example, and compatible results in the other with LINUS and C4.5. It gave similar results to the trained
ANN in both cases and thus, fidelity was also high. In terms of comprehensibility, M-of-N generated significantly smaller rule sets than SUBSET. It also gave in general larger rule sets with more conditions per rule than C4.5, LINUS and EITHER.

2.4.2 Setiono & Liu (1995)

Setiono & Liu (1995) attempted to interpret the predictions given by an artificial neural network by clustering the hidden unit activation values into groups and extract rules by examining the possible combinations in the network output. The algorithm Setiono and Liu proposed in this study was a three-phase algorithm. The first phase of the algorithm is to train a weight decay backpropagation neural network. The trained network is then pruned according to the magnitude of weights, while maintaining prediction accuracy. In the last phase, the training examples are clustered according to their activation values in each of the hidden units and rules can be extracted by examining the possible combinations. The author examined the performance of the proposed algorithm by applying it to three UCI data sets and results were compared to rules generated by C4.5 and C4.5 rules. Results showed that the NN rules had higher accuracies than the decision tree rules but the two sets of rules were quite comparable in terms of understanding measures. The rules generated by the NN rule extraction algorithms were more complex, with more conditions, but the size of the rule sets were generally smaller. One of the disadvantages of the NN rule extraction algorithm pointed out by the authors was the time required for training. However, if time was not a concern, the algorithm presented by the authors provided an insight into how the ANN algorithm works by
showing how the inputs mapped to the hidden units and how those units were mapped to the output units via rules. Fidelity was not considered in this study.

2.4.3 Setiono & Liu (1997)

Setiono and Liu’s previously proposed algorithm was simple and showed desired outcome, however, the algorithm was only suitable for datasets with binary inputs. To overcome this, Setiono and Liu presented another algorithm called NeuroLinear which extracted oblique decision rules, which the decision boundaries are formed by oblique planes rather than axis-parallel plane. The architecture of the ANN in this algorithm is a single layer feedforward neural network with a hyperbolic tangent function and a linear function as the activation function for the hidden and output layer respectively. NeuroLinear was a three-phase algorithm as well. First a neural network with the specified architecture is trained. Then the network is pruned according to the magnitudes of the connection weights. Finally, the hidden unit activation values are discretized using the Chi2 algorithm, which discretizes values based on the $\chi^2$ statistic and rules extracted using the X2R algorithm proposed by the authors. The algorithm was applied to 5 different UCI dataset, with both discrete and continuous attributes, and the results were compared to rules generated by C4.5. The results showed that while the accuracy of the two sets of rules were close, NeuroLinear generated significantly less rules than C4.5 (in best case, it was 3 rules verses 15 rules). Fidelity was also not considered in this study.
2.4.4 Gupta, Park, & Lam (1999)

Gupta, Park, & Lam (1999) proposed the Generalized Analytic Rule Extraction (GLARE) algorithm, which used the connection weights from the trained neural network to extract rules. In the study, the authors explained their reasoning for their proposed algorithm following the “Input-Network-Training-Output-Extraction-Knowledge” framework with three major practices. The first practice is the computational complexity. The author described the approach presented in Setiono and Liu (1995, 1997) and, Kim and Lee (2000) as “generate and test technique”, in which the relationships between input and hidden nodes and, hidden and output nodes are examined separately and the final relationships between input and output are presented by combining the two. The author pointed out that this approach is computation complex, thus they chose the analytic approach in which knowledge are extracted by directly interpreting the connection weights’ strength. The second practice is the specialized network structure and training methods. This practice concerns whether the trained neural network is designed to customize to a specific problem domain or to facilitate the rule extraction process. Such customization causes a tradeoff between simplicity of the rule extraction process and generalization ability of the algorithm. In this study, the authors chose the generalization ability over simplicity. The last practice is the existence of hidden nodes in the final rule sets, in which the authors preferred not to include.

The GLARE algorithm involves a feedforward backpropagation neural network with a single hidden layer. Any continuous attributes are required to be converted to nominal and then Boolean attributes. By ranking the inputs according to their absolute connection
weights to the hidden nodes and importance of each hidden nodes to the considered output, the relative influence of each input are identified and symbolic rules can be extracted. The algorithm was tested on six UCI dataset and the results showed that the GLARE algorithm gave higher prediction accuracies then a neural network algorithm and the C4.5 decision tree algorithm.

2.4.5 Krishnan, Sivakumar, & Bhattacharya (1999)

Krishnan et al. (1999) proposed the COMBO algorithm, in which rules are extracted from a trained neural network by finding the combinations of the input values weights to a neuron that result in an activation value near 1 (for a confirming rule) or near 0 (for a disconfirming rule). Inputs and outputs of the feedforward backpropagation neural network are both assumed to be Boolean and the activation function is the sigmoid function with a high slope so that the activation values will be near 0 or 1. The COMBO algorithm generates rules by finding the significant combinations of connections weights for each neuron. The authors demonstrated their algorithm on two examples and concluded that their algorithm could generate valid and maximally general and complete rules. However, they had also commented that the algorithm had an exponential complexity and would only be suitable for small and medium size neural network.

2.4.6 Kim & Lee (2000)

Kim & Lee (2000) proposed a hybrid technique by combining neural networks and knowledge based system for data classification. As explained by the authors, neural network is used to overcome the shortfall of decision tree, including difficulty in
obtaining the relation between continuous-valued data points; over-generalization for small dataset and over-specialization for large dataset. A feedforward backpropagation neural network with two hidden layers is trained. Both layers use the sigmoid function as their activation function, with high slope values so that each hidden neurons was close to the property of a digital logic gate (with binary value of 0 or 1). The neurons of the first hidden layer are then replaced by logic gates and passed Boolean logic values to the second layer. By doing so, the first hidden layer becomes a linear classifier for the input attributes and their weights. The second and output layer then becomes a sum of product of Boolean logic gates. The algorithm was tested on seven different UCI dataset. The linear classifier obtained from this algorithm was compared to rules obtained by C4.5 and C4.5 plus the linear classifier. The results showed similar accuracies for all three algorithms but much less rules were given by the algorithm proposed by the authors.

2.4.7 Setiono & Leow (2000)

Setiono & Leow (2000) proposed a fast rule extraction algorithm from neural network, called FERNN, which does not require pruning and hence no retaining of the neural network, and reducing computational cost. For the FERNN algorithm, the authors proposed a network training algorithm that “minimizes a cross-entropy error function augmented by a penalty function”. This algorithm will give very small weighs to irrelevant inputs. A decision tree algorithm is then generated based on the activation values of the network’s hidden units and relevant network inputs can be identified. Irrelevant connections between input and hidden neurons are removed with the criterion of maintaining classification accuracy. Finally, rules are given by rewriting the decision
tree node splitting conditions in terms of the network inputs with the inverse sigmoid function. The algorithm was applied to fifteen UCI dataset and results were compared to the rules generated by C4.5 and the combination of N2P2F, a neural network pruning algorithm (Setiono, 1997) and C4.5. The results showed that all three algorithms had similar classification accuracy but the FERNN and N2P2F+C4.5 algorithm gave much smaller rule sets with more complex rules. The authors commended that their algorithm was more general than other rule extraction algorithms and could be applied to both classification and continuous attributes, with high degree of fidelity.

2.4.8 Setiono, Leow, & Zurada (2002)

In this study, Setiono et al. (2002) proposed a rule extraction algorithm, called Rule Exraction from Function Approximating Neural Networks (REFANN), for regression problem. The idea of REFANN is to approximate the hidden neuron activation functions by piecewise linear functions. The piecewise linear function can either be three- or five-piece and the function is approximated based by minimizing the area it covers. The authors assumed the neural network to be a single layer, single output, feedforward backpropagation network with a hyperbolic tangent function as the hidden layer activation function. The REFANN algorithm consists of the following steps: (1) Train and prune a network with one hidden layer and one output; (2) Approximate the hidden activation function with either a three- or five-piece linear function; (3) Rewrite the approximated functions into rules in terms of the corresponding rule conditions; (4) (Optional) Apply C4.5 to simplify the rule conditions. A pruning algorithm called NN pruning for function approximation (N2PFA) was also proposed by the authors. The goal
of the pruning algorithm is to remove less important input and/or hidden neurons, and connections, while maintaining prediction accuracies given by the network.

The authors applied N2PFA on 32 dataset from the WEKA project (Hall et al., 2009) and results were compared to Naïve Bayes (NB), linear regression (LR), k-nearest neighbour (kNN) and a regression tree called, M5’. The neural network after pruning performed better in terms of RMSE than NB, LR and kNN and gave compatible results to M5’. Then the authors applied REFANN to the pruned network. The performance between the pruned network, rules given by a three-piece approximation, five-piece approximation and M5’ were compared. The results showed that the pruned network, and rules given by the five-piece approximation gave compatible performance and both performed slightly better than M5’, while rules given by the three-piece approximation gave close but the lowest accuracies. Although the five-piece approximation approach performed better than the three-piece approximation approach, their performances were both still close to that given by the pruned network, thus both of their fidelities were high. In terms of rule comprehensibility, the authors only examined the number of rules given by their rule extraction algorithms and the five-piece approximation approach gave in general twice as many rules as the three-piece approximation approach.

2.4.9 Setiono & Thong (2004)

In this study, an approach, similar to REFANN (Setiono et al., 2002) was proposed by Setiono & Thong, (2004), where rules are extracted by approximating the hidden neuron activation functions by piecewise linear functions. The differences between this proposed
algorithm and REFANN is that the piecewise linear functions are approximated based on the average of all input and output values of the hidden unit activation function, and the authors only adopted the three-piece approximation approach in this study. The steps in this proposed algorithm is similar to REFANN with the same network architecture requirement as REFANN: a single hidden layer feedforward backpropogation network with hyperbolic tangent function as the hidden activation functions. The steps of the proposed algorithm involves: (1) Training and pruning of the neural network. The pruning algorithm, N2PFA proposed in Setiono et al. (2002) is used; (2) Approximate the hidden layer activation functions with three-piece linear functions; (3) Generate rules from the piece-wise linear functions; (4) (Optional) Apply decision tree algorithm, C4.5, to simplify the rule conditions. The authors applied the proposed algorithm to five UCI datasets and results, in terms of mean absolute errors (MAEs), were compared to three regression tree algorithms KRTree, kNNTrees and LinearTrees (Torgo, 1997) and RUDE (Relative Unsupervised Discretization for regression problems) (Ludl and Widmer, 2000). The results showed that the rules extracted from the neural network model outperformed all of those algorithms. The authors showed the size of the rule set and number of attributes generated by the rule extraction algorithm. However, no comparison were made to other algorithms.

2.4.10 Odajima, Hayashi, Tianxia, & Setiono (2008)

Odajima et al. (2008) proposed a rule extraction algorithm called Greedy Rule Generation (GRG) algorithm, which searches for the best rule that covered the maximum number of samples, the size of subspaces and the number of attributes in each iteration.
This algorithm can be applied to pruned neural network for classification problems. This algorithm is similar to NeruoLinear proposed by Setiono & Liu (1997) and rules are generated by clustering hidden activation values. The proposed GRG algorithm is a sequential covering algorithm, in which one rule is generated at a time and all the examples covered by that rule are removed before the next rule is generated until all examples are being covered. The algorithm was applied to four UCI dataset and the results were compared to those generated by NeuroLinear and C4.5. The results showed that all three algorithms had comparable classification accuracy but the GRG algorithm gave significantly less number of rules. However, as commented by the authors, the algorithm is not suitable for data sets with many attributes and/or attribute values, as the search subspace became too large. The authors suggested some possible solutions including applying feature selection algorithm to select the most relevant attributes and/or values.

2.4.11 Wang et al. (2011)

In this study, Wang et al. (2011) proposed a decompositional rule extraction algorithm for regression problem based on piecewise linear approximation. Similar to REFANN (Setiono et al., 2002) and the algorithm proposed by (Setiono & Thong, 2004), the algorithm extract rules by approximating the hidden neuron activation functions by piecewise linear functions. The authors argued that the approaches employed by REFANN and Setiono & Thong’s (2004) algorithm for solving the piecewise linear approximation did not generate accurate enough results and the least squares approach was used in this study. The network architecture is assume to be a feedforward
backpropagation neural network with one hidden layer with hyperbolic tangent function as the hidden neurons activation functions. The authors demonstrated their algorithm on one set of UCI data and results in terms of prediction accuracies were compared to that given by REFANN, Setiono & Thong’s (2004) and linear regression. The result of the one dataset showed that the algorithm proposed by the authors gave higher accuracy than the other three algorithms. Neither fidelity, nor comprehensibility of the generated rule set was not examined.

2.5 Pedagogical Rule Extraction Algorithm

2.5.1 Saito & Nakano (1988)

Saito & Nakano (1988) proposed a medical diagnostic expect system based on the Parallel Distributed Processing (PDP) model. This is one of the earliest pedagogical rule extraction algorithms and the idea of it is similar to sensitivity analysis. The PDP model is a three layer neural network, with 216 input neurons, 72 hidden neurons and 23 output neurons. Each input neuron represents one possible response to questions regarding the patient’s own perceived symptoms. The outputs are indicators, value ranging from 0 to 1, of the potential relationship between a particular disease and the list of given symptoms. By systematically switching the input neurons to “on” or “off”, the relationship between the symptoms and potential illness can be observed and rules could be extracted. The extracted rules were verified by medical doctors and were considered reasonable. The authors concluded that with 300 training cases, the system showed “almost equivalent” diagnostic capabilities as a symbolic expert system and they expected the performance to further improve if it was further trained with several thousands more cases.
2.5.2 Thrun (1993)

Thrun (1993) proposed a pedagogical rule extraction algorithm based on Validity Interval Analysis (VIA or VI-Analysis), which maps the inputs/outputs relationship in similar manner to sensitivity analysis. The validity intervals constraints the values of the inputs to the activations of the outputs. The intervals are iteratively refined by “detecting and excluding activation values that are logically inconsistent with the weights and bias of the network.” These intervals can be translated into rules which will be in the form of “if input belongs to interval [a, b], then the output will be class 1.” There is no constraint on the type of underlying neural network. The author demonstrated the algorithm on the XOR problem, the Monk’s problem and the robot arm kinetic problem. The results were not compared to any algorithms. Based on the results, the author concluded that VIA is capable to generate “provably correct” rules.

2.5.3 Craven & Shavlik (1996)

Craven & Shavlik (1996) proposed a pedagogical rule extraction algorithm, called TREPN, which uses the trained neural network as an oracle to construct a decision tree. The given decision tree is expected to describe the concept represented by the trained neural network. TREPN is similar to conventional decision-tree algorithms, such as CART (Breiman et al, 1984) and C4.5 (Quinlan, 1993), but with three major differences:

1. TREPN grows trees best-first instead of depth-first. As explained by the authors, this increases the fidelity of the extracted tree.

2. TREPN uses m-of-n expressions for splitting instead of a single feature.
3. While the depth of conventional decision-tree is limited by the amount of training data, TREPAN overcomes this limitation by generating additional data using the oracle. Thus, TREPAN is able to consider at least a minimum number of data before making a split.

Craven & Shavlik (1996) demonstrated their algorithm on four UCI dataset and evaluated the results based on the rules predictive accuracy, comprehensibility and fidelity to the neural network. The accuracies of the extracted rules were compared to those of the trained neural network and rules given by the C4.5 algorithm and ID2-of-3 algorithm (Murphy & Pazzani, 1991). The results showed that while neural networks still gave the most accurate results, TREPAN gave significantly better results than C4.5 and ID2-of-3. The accuracies of the extracted decision tree were only slightly lower than that given by the neural networks, thus fidelity of the extracted tree was high. The comprehensibility of the trees was evaluated based on the number of internal nodes and number of symbols in the rules. The results showed that the TREPAN tree has less number of internal nodes than both of the trees generated by C4.5 and ID2-of-3. While the number of symbols of the TREPAN tree was higher than the C4.5 tree, it was still significantly lower than ID2-of-3.

2.5.4 Huysmans, Baesens, & Vanthienen (2006)

Huysmans et al. (2006) proposed the ITER (latin for “journey”) algorithm which can extract regression rules from neural networks and support vector machines. The ITER algorithm extracts rules by dividing the input space into a number of hypercubes. The algorithm starts with the user defining the number of starting cubes. During each
iterations the algorithm expands the cubes, until they cover the entire input space. The authors applied ITER to datasets with target values generated by the least-squares support vector machines (LS-SVM) algorithm in order to mimic the behavior of the given algorithm through rule extraction. The results were compared to that given by Linear Regression (LR), K-Nearest Neighbor (KNN), LS-SVM and CARTS. Although the results showed that CART outperformed ITER slightly on most datasets, the authors argued that ITER had the advantage over CART as a non-greedy approach and would be able to find better rules than CART when CART couldn’t find a split that decreases impurity significantly. The number of rules generated by ITER depends on the problem and the threshold of the LS-SVM. In one of the examples, ITER generated more than twice as many rules than CART with a threshold of 0.10 (12 rules Vs 27 rules) but in another other example, CART generated almost 9 times more rules than ITER with the same threshold (92 rules Vs 11 rules).

2.5.5 Saad & Wunsch (2007)

The algorithm proposed by Saad & Wunsch (2007) was called HYPINV which stands for HYPerplanes using INVersion. As suggested by the name, the algorithm finds the hyperplanes that correspond to the trained neural network decision boundary by the neural network inversion technique. The goal of the neural network inversion technique is to find the sets of input correspond to a given target output. The idea of the inversion technique is similar to backpropagation in which the error signals are propagated back to adjust the network, but instead of adjusting the weights, the inversion algorithm adjusts the activation values of the input units while the network weights are frozen (Linden &
Kindermann, 1989). The HYPINV algorithm uses this technique to find the hyperplanes of the neural network. In a two-class decision problem, the hyperplanes divides the inputs into positive and negative subspaces and classifications of the output are made accordingly. The found hyperplanes can be expressed into conjunction and disjunction rules. The authors demonstrated their algorithm on six UCI datasets and results were compared to NeuroLinear (Setiono & Liu, 1997) and Kim & Lee's (2000) algorithm. The results showed that HYPINV gave slightly lower but compatible accuracies than NeuroLinear and Kim & Lee's (2000) algorithm, however, HYPINV gave much less rules. The fidelities of the proposed algorithm were high and was in the range of 88-99%.

2.5.6 de Fortuny & Martens (2012)

de Fortuny & Martens (2012) proposed a pedagogical rule extraction algorithm based on active learning. The ALPA-R (Active Learning-based Pedagogical Rule Extraction for Regression) algorithm can be applied to any “black-box” algorithm, such as artificial neural network and support vector machines. The ALPA algorithm trains a “white-box” model by an induction technique, using the predicted target values given by the “black-box” model. The idea is similar to that in TREPAN (Craven & Shavlik, 1996) and the trained “black-box” is used as an oracle. Active leaning is used to train the “white-box” model, in which the algorithm chooses the input data points it wants to get the label for any given point based on its query strategy. The authors implemented the ALPA-R algorithm in WEKA and users can apply the algorithm on any built-in non-transparent (“black-box”) and rule induction models in WEKA. The authors demonstrated their algorithm on the several UCI data sets, by using support vector machine regression
(SVR) as the “black box” model and REPTree, a regression tree algorithm, as the “white-box” model. Results showed that the ALPA-R algorithm performed better in both accuracy and fidelity than the original REPTree algorithm. The authors concluded that although the ALPA-R algorithm showed adequate level of fidelity and accuracy, they believed there was room for improvement and in order to do so, the query strategy by the active learning component needed to be further improved.

2.5.7 Augasta & Kathirvalavakumar (2012)

Augasta & Kathirvalavakumar (2012) proposed a pedagogical rule extraction algorithm called RxREN based on reverse engineering. The idea of reverse engineering is to determine the components of a product by studying the finished item. The authors applied this principle to the RxRen algorithm and the algorithm identifies the data range of each significant input neuron to generate a correct output by analyzing the misclassified outputs. These data range can then be translated into rules. This algorithm is suitable for classification problems with either discrete or continuous inputs and its performance was demonstrated on six UCI data sets. The algorithm were also compared to one other pedagogical approach: HYPINV (Saad and Wunsch, 2007), three decompositional approach: NeuroLinear (Setiono & Liu, 1997), Kim & Lee's (2000), and GRG (Odajima et al., 2008) and one eccentric approach: Rx+CGA (Hruschka & Ebecken, 2006). The results show that the extracted rules had comparable accuracies to the original trained neural network. Compare to other algorithms, RxREN extract rules with smaller rule sets then some algorithm but comparable accuracies. On the Wisconsin breast cancer, RxREN generated less rules than NeuroLinear, Kim & Lee’s (2000) and Rx+CGA; same amount
of rules as GRG; but more than HYPINV. The authors commented that the main advantage of the RxREN was its low computational cost while maintaining high accuracy and process of rule pruning also improved the algorithm’s generalization ability.

2.6 Eclectic Rule Extraction Algorithm

2.6.1 Hruschka & Ebecken (2006)

In their research, Hruschka & Ebecken (2006) proposed a rule extraction algorithm called clustering genetic algorithm (CGA). As the name suggested, CGA involves clustering based on genetic algorithm. The idea of clustering hidden unit activation values in order to extract rules from a trained neural network is based on the RX algorithm proposed by Lu, Setiono, & Liu (1996). However, in order to optimize the number clusters, and hence number of rules extracted, Hruschka & Ebecken employed genetic algorithm to perform the clustering. CGA uses the instance corresponds to the most gain as the initial population in order to accelerate the convergence of the algorithm. The authors demonstrated their algorithm four UCI data set and one real-world meteorological dataset and the average correct classification rates (ACCRs) given by the trained neural network, CGA and the algorithms proposed in other literature. The results showed that for the UCI data, CGA had on average better accuracies than the literatures and slightly lower but compatible accuracies to the neural network models. Although not as accurate as the UCI datasets, CGA still gave compatible results to the neural network on the real-world data.
2.6.2 Khosrow & Sandesh (2006)

The algorithm Khosrow & Sandesh (2006) proposed involves clustering of the hidden layer activation values in order to extract rules, similar to the ones proposed by Lu, Setiono, & Liu (1996) and Hruschka & Ebecken (2006). Khosrow & Sandesh's algorithm involved the following steps: (1) neural network training and filtering, (2) pruning and re-training, (3) clustering the hidden neuron activation values, and (4) rule discovery and extraction. After clustering, dataset are fed into the trained and modified neural network, and the control parameters (cluster radius, confidence frequency, and hidden layer activation level) are used to determine if a given hidden layer neuron should be active. An output will be generated if the total number of inactive hidden layer neurons does not exceed the maximum allowed number of inactive hidden layer neurons. This limits the number of input patterns that will generate an output and the authors claimed that the input patterns that pass the “rigorous extraction phase” will produce “an output pattern represent generalization and correlations that exist in the dataset”. The authors demonstrated the performance of the proposed algorithm differently to all the other papers mentioned here. Instead of attempting to classify the given data accurately, the authors wanted to discover the “trend” in the dataset and they showed how much in percentage, the dataset supported the discovered trend. No actual rules were given by the algorithm.

2.6.3 Kahramanli & Allahverdi (2009)

In this study, Kahramanli & Allahverdi (2009) proposed a rule extraction algorithm for neural networks with adaptive activation functions, using an artificial immune system
(AIS) algorithm. The major difference between a backpropagation and adaptive neural network is that the adaptive neural network has an activation function with adjustable variables and is usually in the form of

$$\varphi(x) = A_1 e^{-x^2} + \frac{A_2}{1 + e^{-B x}}$$

where A1, A2 and B are real variables which will be adjusted during training. With the above activation function, the output of the trained ANN, C(X), is given by:

$$C(X) = A_1 e^{-\sum_{i=1}^{n} d_i e^{-\left(\sum_{j=1}^{m} \gamma_{ij} - \gamma_j\right)^{2}}} + A_2 \left(\sum_{j=1}^{m} \left(\sum_{i=1}^{n} \gamma_{ij} - \gamma_j\right)^{-1}\right)^{-1} + \frac{A_2}{1 + e^{-B \cdot X}}$$

The input and output values needed to be encoded, e.g. if the input attribute can be high, medium or low, then that input attribute will be encoded as 100 for high, 010 for medium and 001 for low. Since the output can only be 0 or 1 after encoding, solving the output equation showed above becomes an optimization problem of finding either the maximum or minimum solution. In this study, the authors employed an artificial immune system, called Opti-aiNet (de Castro & Timmis., 2002), to solve the optimization problems. Each of the solutions given represented a rule. The authors demonstrated their algorithm on two sets of UCI data and the rule prediction accuracies were compared to that given by other rule generating algorithms such as CART and inductive algorithm, however, some of the references were not given clearly. The results showed that the rules extracted from an adaptive neural network had higher accuracies than those from a standard neural
network and other rule generating algorithms. No results on fidelity or comprehensibility were given.

2.6.4 Al Iqbal (2011)

Al Iqbal (2011) proposed an eclectic rule extraction algorithm called HERETIC (Hierarchical and Eclectic Rule Extraction via Tree Induction and Combination). The algorithm extracts symbolic rules by approximating each of the neuron of a trained neural network with a decision tree. Since the author assumed that each neuron only has two possible output values, 0 or 1, the algorithm requires the activation function of the neural network to be a steep sigmoid function. Once the decision trees are generated, each of the trees can be expressed in terms of conjunction rules. The conjunction rules are then combined with conjunction rules from other neurons to give the rule sets of the trained neural network. A logical simplification algorithm was employed as the final step to simply the rules into disjunctive normal form. The algorithm was applied to seven UCI datasets and results were compared to the decompositional algorithm FERNN and pedagogical algorithm Trepan. The result showed that rule generated by HERETIC had compatible accuracy as the trained ANN, and outperformed both FERNN and Trepan in terms of fidelity.

2.6.5 Mohamed (2011)

In this study, Mohamed (2011) proposed an algorithm to extract rules from a constructively trained neural network using genetic algorithm. To constructively train a neural network, one starts with a small network and then expanding the network by
adding more hidden units and weights until the neural network gives desired performance. In this study, the author started with one hidden unit and convergence rate and error were checked after each addition of hidden units, until a desired accuracy was reached. Genetic algorithm was used to solve the optimization problem of the output function, similar to the application of artificial immune system in Kahramanli & Allahverdi (2009)’s algorithm. Since the input and output attributes were also encoded like in Kahramanli & Allahverdi (2009), finding the input values corresponding to an output of 0 or 1 was an optimization problem of finding the minimum and maximum solution. The optimization problem could be stated as

$$\text{Maximize } \psi(x) = \frac{1}{1 + e^{-\sum_{i=1}^{n} y_i (1/1 + e^{-\sum_{j=1}^{m} w_{ij}})}}$$

subjected to: $x_i \in \{0 \text{ or } 1\}$.

The author demonstrated the proposed algorithm on four UCI dataset and results were compared to Bagging with a tree based approach, C4.5 with reduced error pruning (REP), Naïve Bayesian, and RBF networks. Results showed that the proposed method gave significantly more accurate results than Naïve Bayesian, REP tree and RBF networks in all the datasets and only Bagging gave compatible accuracies in most of the dataset. The author claimed that the proposed algorithm also gave “more simple and comprehensible rules in comparison with other algorithms” but comprehensibility, such as number of rules and number of attributes in the rules, were not shown.

2.7 Comparisons on the Rule Extraction Algorithms

The rule extraction algorithms reviewed are classified and evaluated based on the criteria explained in Chapter 2 and the results are summarized in Table 2.1 and Table 2.2.
Table 2.1 summaries the translucency, scope, accuracy, fidelity, generality and the expressive power of the given rules of the reviewed algorithms and the following index are used in the table:

- **Translucency**: D, P or E
  (Decompositional, Pedagogical or Eclectic)

- **Scope**: C, R and/or RD
  (Classification, Regression and/or Regression but require Discretization of data)

- **Accuracy**: H, M, L or N/S
  (High= 90-100%, Medium=75-89% or Low=below75%, N/S = not shown)

- **Fidelity**: H, M, L or N/S
  (High=90-100%, Medium=75-89% or Low= below 75%, N/S = not shown)

Generality refers to whether the rule extraction algorithm requires an underlying ANN model with specific training regimes and/or network architecture. Therefore the ANN architecture and any related requirement is noted under generality.

In terms of expressive power, it can be M-Of-N rules, If/Then rules, decision tree and numeric functions.
Table 2.1 – Comparison of rule extraction algorithms on classification and performance criterion

<table>
<thead>
<tr>
<th>Authors/ Name of Algorithm</th>
<th>Transl. Scope</th>
<th>Accuracy</th>
<th>Fidelity</th>
<th>Generality</th>
<th>Expressive Power</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Decompositional Algorithms</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2  Setiono &amp; Liu, (1995) NeuroLinear</td>
<td>D  C  H  N/S</td>
<td>Single hidden layer; hyperbolic tangent function; Pruning was included.</td>
<td>If/Then rules</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3  Setiono &amp; Liu, (1997)</td>
<td>D  C, RD  M-H  N/S</td>
<td>Single hidden layer, hyperbolic tangent function; Pruning was included</td>
<td>If/Then rules</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4  Gupta, Park, &amp; Lam (1999) GLARE</td>
<td>D  C, RD  M-H  H</td>
<td>Single hidden layer</td>
<td>If/Then rules</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5  Krishnan, et al. (1999)</td>
<td>D  C  N/S  N/S</td>
<td>Sigmoid function with high slope</td>
<td>Decision tree, If/Then rules</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7  Setiono &amp; Leow, (2000) FERNN</td>
<td>D  C  H  N/S</td>
<td>Special NN with different penalty function</td>
<td>If/Then rules or M-of-N rules</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Author(s)</td>
<td>Method</td>
<td>Layer Type</td>
<td>Activation Function</td>
<td>Pruning Required</td>
</tr>
<tr>
<td>---</td>
<td>-----------------------------------</td>
<td>-----------------------</td>
<td>------------</td>
<td>---------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>10</td>
<td>Odajima, Hayashi, Tianxia, &amp; Setiono, (2008), GRG</td>
<td>D C L-H N/S</td>
<td>Pruned NN</td>
<td>If/Then rules</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Wang, Qin, Zhang, &amp; Shi  (2011)</td>
<td>D R H N/S</td>
<td></td>
<td>Single hidden layer, hyperbolic tangent function, pruning required.</td>
<td>Numeric Function</td>
</tr>
</tbody>
</table>

**Pedagogical Algorithms**

<table>
<thead>
<tr>
<th></th>
<th>Author(s)</th>
<th>Method</th>
<th>Layer Type</th>
<th>Activation Function</th>
<th>Pruning Required</th>
<th>Decision Rule Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Saito &amp; Nakano, (1988)</td>
<td>P C N/S N/S</td>
<td></td>
<td>PDP model</td>
<td>If/Then rules</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Thrun (1993)</td>
<td>P C H H</td>
<td>Any trained NN</td>
<td>If/Then rules</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Huysmans, Baesens, &amp; Vanthienen, (2006) ITER</td>
<td>P R H H</td>
<td>Trained NN or SVM</td>
<td>If/Then rules</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Saad &amp; Wunsch, (2007) HYPINV</td>
<td>P C L-H H</td>
<td>Any trained NN</td>
<td>If/Then rules</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Authors</td>
<td>Algorithm</td>
<td>Initialisation</td>
<td>H</td>
<td>Pruning</td>
<td>Rule Set</td>
</tr>
<tr>
<td>---</td>
<td>--------------------</td>
<td>-----------</td>
<td>----------------</td>
<td>---</td>
<td>---------</td>
<td>----------------</td>
</tr>
<tr>
<td>7</td>
<td>Augasta &amp; Kathirvalava kumar (2012) RxRen</td>
<td>P C M-H H</td>
<td>Pruning is included</td>
<td>If/Then rules</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Eclectic Algorithms</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Hruschka &amp; Ebecken, (2006) CGA</td>
<td>E C H H</td>
<td>Any trained NN</td>
<td>If/Then rules</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Kahramanli &amp; Allahverdi (2009)</td>
<td>E C H N/S</td>
<td>Adaptive NN, but the AIS approach could be applied to any trained NN</td>
<td>If/Then rules</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Al Iqbal, (2011) HERETIC</td>
<td>E C H H</td>
<td>Steep sigmoid function</td>
<td>If/Then rules</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Mohamed, (2011)</td>
<td>E C H N/S</td>
<td>Constructively trained NN; but the GA approached could be applied to any trained NN.</td>
<td>If/Then rules</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Since we are also interested in the application of ANN rule extraction algorithms, particularly in the field of engineering, we also analyze the datasets that were used in the previous studies. Table 2.2 summaries the datasets that were used in the studies.
Table 2.2 – Comparison of rule extraction algorithms on dataset used and rules comprehensibility

<table>
<thead>
<tr>
<th>Authors/ Name of Algorithm</th>
<th>Dataset used</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Decompositional Algorithms</strong></td>
<td></td>
</tr>
</tbody>
</table>
2. Splice-junction determination |
| 2  Setiono & Liu, (1995) | 1. Iris  
2. Breast cancer  
3. Splice juice |
| 3  Setiono & Liu, (1997) | 1. Australian credit approval  
2. Boston housing  
3. Cleveland heart disease  
4. Wisconsin breast cancer  
5. Sonar target |
| 4  Gupta, Park, & Lam (1999) | 1. Postoperative Patient  
2. Balloon  
3. Hepatitis  
4. BUPA liver-disorders  
5. Glass  
6. Iris |
| 5  Krishnan, et al. (1999) | 2 simple examples, including encoder problem |
2. Iris  
3. Wisconsin breast cancer  
4. Ionosphere  
5. Pima Indians diabetes  
6. Glass  
7. BUPA liver-disorders |
| 7  Setiono & Leow, (2000) | 1. Monk 1  
2. Monk 2  
3. Monk 3  
4. Australian credit approval |
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Setiono, Leow, &amp; Zurada, (2002) REFANN</td>
<td>32 datasets from the weka project: 4 are engineering related</td>
</tr>
<tr>
<td>11</td>
<td>Wang, Qin, Zhang, &amp; Shi (2011)</td>
<td>Computer hardware</td>
</tr>
<tr>
<td><strong>Pedagogical Algorithms</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Saito &amp; Nakano, (1988)</td>
<td>Medical System</td>
</tr>
<tr>
<td></td>
<td>Huysmans, Baesens, &amp; Vanthienen, (2006)</td>
<td>4 artificial dataset, including a linear function and a parabolic function</td>
</tr>
<tr>
<td></td>
<td>Authors</td>
<td>Datasets</td>
</tr>
<tr>
<td>---</td>
<td>----------------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
2. Diabetes  
3. Boston housing data  
4. Machine CPU  
5. Pyrim  
6. Servo  
7. Stock  
8. Wisconsin breast cancer |
| 7 | Augasta & Kathirvalavakumar (2012) RxRen | 1. Iris  
2. Wisconsin breast cancer  
3. Pima Indians diabetes  
4. Hepatitis  
5. German credit card  
6. Ionosphere |

**Eclectic Algorithms**

<table>
<thead>
<tr>
<th></th>
<th>Authors</th>
<th>Datasets</th>
</tr>
</thead>
</table>
| 1 | Hruschka & Ebecken, (2006) CGA    | 1. Iris  
2. Australian credit approval  
3. Pima Indians diabetes  
4. Wisconsin breast cancer  
5. Meteorological |
| 3 | Kahramanli & Allahverdi (2009)   | 1. Ljubljana breast cancer  
2. ECG |
| 4 | Al Iqbal, (2011) HERETIC         | 1. Promoters  
2. Wisconsin breast cancer  
3. Congressional voting  
4. Heart disease  
5. Monk 1  
6. Monk 2  
7. Monk 3 |
| 5 | Mohamed, (2011)                  | 1. Monk  
2. Wisconsin breast cancer |
Based on the literature review and evaluation, the following observations can be made regarding the neural network rule extraction literature reviewed:

**Observation 1:** There is a trade-off between the accuracy and the complexity of the neural network.

This observation is mostly concerned with neural network pruning. Some of the algorithms, such as Setiono & Liu (1995), emphasized the importance of network pruning. The purpose of network pruning is to decrease the complexity of the neural network by removing less significant connections and/or neurons. By doing so, the generalization ability of the network as well as the comprehensibility of the generated rules (Khosrow & Sandesh, 2006) is improved. However, Setiono & Liu's (1995) study also demonstrated the trade-off between the accuracy and the complexity of the neural network. The less complex neural network can give less accurate predictions. And assuming the rule extraction algorithm has a high fidelity to the neural network, network pruning will also decrease the accuracy of the generated rule. Thus, a balance between the neural network complexity and accuracy is needed.

**Observation 2:** There is a trade-off between the fidelity and comprehensibility of the extracted rules.

This observation echoes the first suggestion given by Craven & Shavlik (1999) on improving the scalability of the algorithms. A trained neural network that is highly accurate most likely means that it is a complex neural network, and if we are to
implement a rule extraction algorithm that mimics the exact behavior of the neural network, the generated rules are likely to be complex with multiple conditions. An example can be given by Setiono et al.’s study (2002), in which the authors proposed an algorithm that approximates the hidden neuron activation functions with either a three-piece or five-piece linear function. While the five-piece approach gave higher fidelity than the three-piece approach (in the range of 98 - 100% Vs 91 – 100%), they gave on average almost twice as many rules as the three-piece approach. Thus, there is a clear trade-off between the two properties of fidelity and comprehensibility.

**Observation 3:** There is a trade-off between rule-set size and the number of attributes in the generated rules.

Most of the studies reviewed do not evaluate the rule comprehensibility. However, the ones that do usually show that the rule extraction algorithms generate a smaller rule set with compatible accuracy to traditional rule generating algorithm such as C4.5. This could be due to the fact that most of the NN rule extraction algorithm gives more complex rules with more conjunctions of conditions. For example, in Setiono & Liu (1995), among all three of the datasets the authors tested their algorithm on the rule sets given by the ANN rule extraction algorithm had 25-50% fewer rules than the rule sets given by the decision tree algorithm. However, the average number of conditions per rule given by the ANN rule extraction algorithm was 20 – 45% higher than the decision tree algorithm. Since some of the conjunctions are combined, the number of rules has decreased and this issue is often over looked. Rules comprehensibility is a major problem in many of the algorithms. In this study, we plan to generate rule sets that have a balance
between the rule set size and number of conditions per generated rule in order to maintain comprehensibility.

**Observation 4:** Scalability of the algorithm is usually not examined.

As pointed out by Craven & Shavlik (1999), the scalability of the rule extraction is always overlooked. And even when scalability is examined, the results are usually not desirable. For example, the concept of the COMBO algorithm (Krishnan et al., 1999) was quite straightforward; however, the authors commented that the exponential complexity of their algorithm made it not suitable for real problems. Similar comment was also made by Odajima et al. (2008) on their GRG algorithm. Thus, we planned to examine the scalability of our algorithm by applying it to more complex ANN and datasets. This refers to ANNs with large number of inputs and hidden neurons.

**Observation 5:** There is a lack of algorithms for regression problems.

Of all the algorithms reviewed, only a handful of them were designed for regression problems (*italic* in Table 2.1). This is an area worth investigating as many real world problems, such as those involving predictions, require some crisp numeric values rather than the ranges of values within which the predicted outcome possibly lie.

**Observation 6:** Pedagogical approaches have an advantage in generality over decompositional approaches.

This has been pointed out by Craven & Shavlik (1999), and our review can further confirm this. Usually, the decompositional approaches specified the topology and/or
architecture of the neural network, e.g. a steep sigmoid function (Krishnan, et al., 1999), neural network with special penalty function (Setiono & Leow, 2000). By contrast, most of the pedagogical approach does not involve special implementation constraints and some of them could even be applied to other black-box algorithm apart from ANN, such as support vector machines, e.g. ALPA-R (de Fortuny & Martens, 2012) and ITER (Huysmans et al., 2006). Hence, the pedagogical approach is usually considered to be more general.

**Observation 7:** There is a lack of applications on engineering datasets.

As shown in Table 2.2, almost all the algorithms were tested on UCI datasets and a few engineering datasets, which are underlined. Sometimes these datasets are rather straightforward and simple and would not be able to reflect the ability of the proposed algorithm on engineering problems that are usually more complex. Therefore, we will apply our developed algorithm to datasets from engineering problems.

**Observation 8:** The online availabilities of developed algorithms in the public domain are very limited.

### 2.8 Multiple Linear Regression

Multiple linear regression (MLR) is a statistical technique that linearly models the relationships between multiple predictor variables, \( x_n \), and a single predicted variable, \( y \); it usually has the following form:

\[
y = a_1 x_1 + a_2 x_2 + ... + a_n x_n + b
\]
where \( y \) is the predicted variable, \( a_1 \) is the coefficient of the predictor variable \( x_1 \), \( a_2 \) is the coefficient of the predictor variable \( x_2 \), \( a_n \) is the coefficient of the predictor variable \( x_n \), and \( b \) is a constant variable that represents the difference between the predicted and observed value of \( y \).

The MLR approach is based on four assumptions (Poole & O’Farrell, 1971; Osborne & Waters, 2002; Uyanık & Güler, 2013):

1. Assumption of linearity: The predictor and predicted variables are assumed to be linearly related, which can be verified by examining the scatter plots of each of the predictor variables against the predicted variable.

2. Assumption of normality: The predictor variables or residuals of the model are assumed to be normally distributed, which can be seen from the residual normality probability plot.

3. Assumption of no or little multi-collinearity: This assumption requires that there should be no or little correlation between two or more predictor variables. If multi-collinearity exists, the variance inflation factor (VIF) and tolerance can be used to measure multi-collinearity (Field, 2000).

4. Assumption of homoscedasticity: For all of the predictor variables, the variance around the regression line is assumed to be the same. This assumption can be verified by examining the residual plot, which is a scatter plot of standardized residuals versus the predicted values.
While there are various methods, such as the least absolute deviations, the least squares deviations, and minimax absolute deviations (Kumar & Singh, 2015), that can be used to fit the MLR models, the least squares method is the most widely used among them (Abdi, 2007). The least squares method estimates the coefficients of the predictor variables by minimizing the sum of the squared deviations or residuals between the predicted and observed y values. Assuming the predicted values are given by \( \hat{y}_i \), and the observed values are given by \( y_i \), then the objective function of the least squares deviation method is given by

\[
\min |\hat{y}_i - y_i|^2
\]

The statistical significance of the solved MLR equation and of each of the predictor variables can be evaluated by the values of F-ratio and t-statistic, respectively (Zhou et al., 2009). The F-ratio tests the hypothesis that at least one of the coefficients given by the MLR equation is significant, i.e. not zero. The value of F-ratio is given by

\[
F = \frac{R^2 \times (N - K - 1)}{(1 - R^2) \times K}
\]

where \( R^2 \) is the multiple correlation coefficient, \( K \) is the number of predictor variables, and \( N \) is the number of data tuples in a given dataset. A larger value of F indicates that there is a stronger relationship between the predictor variables and the predicted variable, and a value of F that is greater than one is desired.

The t-statistic, on the other hand, tests the significance of each coefficient. The value of t is given by,

\[
t = \frac{a_t}{S_{at}}
\]
where $a_t$ is the regression coefficient of the predictor variable $t$, and $S_{at}$ is the standard error of the respective coefficient. The null hypothesis that the specified coefficient is insignificant can be rejected if the value of $t$ is greater than the pre-defined critical value, $t_c$. 

CHAPTER 3 PRELIMINARY METHODOLOGY – THE PWL-ANN ALGORITHM

In this chapter the development of a decompositional neural network rule extraction algorithm, called PWL-ANN, for non-linear regression problems is presented. The approach adopted is to model a given dataset using the ANN approach and the originally trained neural network is assumed to be a three-layer feed-forward backpropagation neural network with a sigmoid activation function. Since these are the most common types of neural network models, this will be the target models on which rule extraction will be performed. Also, based on experience, one hidden layer in an ANN is typically sufficient to solve most non-linear problems without overfitting. Although the pedagogical type of rule extraction algorithms is better in terms of computational complexity and generality than decompositional algorithms, the decompositional approach is the focus because the objective is to explore the trained neural network and “open the black box.” The algorithm approximates the activation functions of a given ANN model with piece-wise linear (PWL) equations and generates explicit information in the form of numerical formulae. The targeted problems are regression problems related to engineering domains. In terms of expressive power, we would like to generate “rules” expressed as linear numeric functions in the form of

\[ y = a_1 x_1 + a_2 x_2 + \ldots + a_n x_n + b \]

Since the research objective is to understand the working mechanism of the trained neural network model, fidelity to the trained ANN will be the primary evaluation criterion of the developed algorithm. The performance of the proposed algorithm was evaluated using 15 datasets obtained from the UCI repository (Lichman, 2006) and 4 datasets obtained from
the post-combustion carbon dioxide capture process system implemented at the Clean Energy Technology Research Institute (CETRI) located at the University of Regina, in Regina, Saskatchewan, Canada.

3.1 Methodology

The PWL-ANN algorithm is built on two concepts. First, the piece-wise linear (PWL) algorithm is responsible for approximating the hidden activation functions of a trained neural network. Secondly, the neural network with the hidden layer replaced by the generated PWL equations is called the piece-wise linear neural network (PWL-ANN). Rapidminer (trademark of Rapidminer) has been used to train the neural network, and the rule extraction algorithm is written in R (trademark of R). The main steps of the PWL-ANN algorithm are as follows:

1. Training the neural network for a given dataset.

The data is first normalized to the range of [-1, 1] before training and Rapidminer’s neural network model (trademark of Rapidminer) was used to model the dataset. The model uses a sigmoid function and a linear function as the activation functions for the hidden layer and the output layer, respectively. The training cycles were set to 500 with a learning rate of 0.3 and a momentum of 0.2; both are default settings. Only a single hidden layer is considered, and the number of nodes in the hidden layer by default in Rapidminer is given by (number of input attributes + number of output attributes)/2 + 1. The model is trained with a 20-fold cross-validation, and Rapidminer provides information on the trained ANN model and its performance in terms of the root mean
squared error. The weights of the nodes of the trained model given by Rapidminer are used in the PWL approximation algorithm.

2. Approximating the activation functions of the hidden neurons of the trained ANN model using the three-piece linear (3PL) functions.

\[ f(\xi) = \frac{1}{1 + e^{-\xi}} \]

where \( \xi \) is the sum of the weighted input:

\[ \xi = \sum_{i=1}^{n} w_i x_i + \theta \]

where \( w_i \) are the weights and \( \theta \) is the bias.

The sigmoid function can be approximated using a three-piece linear function, \( L(\xi) \); such that:

\[
L(\xi) = \begin{cases} 
  a_1 \xi + b_1, & \text{for } \xi > \xi_0 \\
  a_0 \xi + b_0, & \text{for } -\xi_0 \geq \xi \geq \xi_0 \\
  a_1 \xi - b_1, & \text{for } \xi < -\xi_0 
\end{cases}
\]

As shown in Figure 3.1, \( \xi_0 \) and \(-\xi_0\) are called the breakpoints (BPs) of the piece-wise linear equations. The challenge of finding an accurate 3PL equation lies in locating the breakpoints, which can be found by performing a brute-force search on the dataset. The steps involved in approximating the activation functions are as follows:

Let \( M \) be the total number of data points in a given dataset,

i. Sort the data points in ascending order.
ii. Divide the dataset into $M/d$ number of subsets, where $d$ is the number of data points per subset, and this value determines how frequently the data points within the dataset are tested. Each tested point becomes a potential breakpoint, in other words, the breakpoints are the points at which the function changes. For example, $d=1$ means each point will be tested, and $d=10$ means every 10 points of the dataset will be tested.

iii. For each subset, consider the last data entry of the subset, $\xi_i$, as the breakpoint. Solve the linear equations for the data entries in the subintervals of $[0, \xi_i]$ and $[\xi_i, \xi_{max}]$ using the least squares approach. Record the mean squared error (MSE) given by the two lines.

iv. Repeat step (iii) with all the subsets.

v. The set of equations with the lowest value of MSE is considered the best solution.

Due to the symmetry of the sigmoid function, it is only necessary to locate one breakpoint, $\xi_0$, and the other breakpoint will be $-\xi_0$. Based on this constraint, two possible approaches can be adopted to solve the 3PL equations:

a) Either the dataset that lies on the left, i.e. negative $\xi$ values, or right, i.e. positive $\xi$ values, hand side of the $y$-axis is used, the choice favors the side that has the larger set of data.

b) The absolute values of $\xi$ are used to solve the linear equations, and it is necessary to locate only one breakpoint, $\xi_0$. The $L(\xi)$ values of negative $\xi$ can be obtained by $1 - L(\xi)$. 


Figure 3.1 – Approximating the activation function (sigmoid function) with a three-piece linear approximation function.
3. Extracting rules from the approximated linear functions.

After the activation functions of the hidden neurons are approximated using 3PL equations, the sigmoid functions are replaced with the 3PL functions and the PWL-ANN model is derived. Then rules in the form of linear functions can be extracted from the PWL-ANN model. To illustrate the result of the algorithm, consider a neural network with three inputs and three hidden neurons as shown in Figure 3.2. By approximating the activation functions with linear equations, the output can be expressed in terms of the inputs as a linear equation:

\[
y = w_1(a_1(w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + \theta_1) + b_1) + w_2(a_2(w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + \theta_2) + b_2) + w_3(a_3(w_{31}x_1 + w_{32}x_2 + w_{33}x_3 + \theta_3) + b_3) + \theta_4
\]

which can be simplified as

\[
y = Ax_1 + Bx_2 + Cx_3 + D
\]

where

\[
A = w_1a_1w_{11} + w_2a_2w_{21} + w_3a_3w_{31}
\]

\[
B = w_1a_1w_{12} + w_2a_2w_{22} + w_3a_3w_{32}
\]

\[
C = w_1a_1w_{13} + w_2a_2w_{23} + w_3a_3w_{33}
\]

\[
D = w_1(a_1\theta_1 + b_1) + w_2(a_2\theta_2 + b_2) + w_3(a_3\theta_3 + b_3) + \theta_4
\]

and are all constants. For each of the given input, we can now find the corresponding multiple linear equation that calculates the output. These linear equations become the output “class” of the dataset and if-then rules can be generated based on these classes.
Figure 3.2 – A trained neural network with three inputs, three hidden neurons and one output.
3.2 Results and Analysis

To assess performance of the PWL-ANN algorithm, it was tested on 19 datasets, among which 15 datasets were obtained from the UCI repository (Lichman, 2006) and the Weka project (Hall et al. 2009), and 4 datasets (number 16 – 19) were obtained from the post-combustion carbon dioxide capture process system implemented at the Clean Energy Technology Research Institute (CETRI) located at the University of Regina, in Regina, Saskatchewan, Canada. These datasets were chosen because they all have continuous output values. The characteristics of the datasets are summarized in Table 3.1. For the discrete attributes, dummy variables were used to represent the different classes of that attribute. For example, for an input attribute that has the classes of sunny, cloudy, and windy, with the dummy coding, the classes are labeled as 1, 2 and 3.

Based on the same PWL-ANN algorithm, three approaches to solving the PWL approximation process were adopted:

1. Approach 1: Using the larger half of the dataset with d=10;

2. Approach 2: Using the larger half of the dataset with d=3;

3. Approach 3: Using the absolute values of the dataset with d=3.
### Table 3.1 – Characteristics of the datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Data size</th>
<th>No. of input nodes</th>
<th>No. of hidden nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1 Energy Efficiency Heat loading (UCI)</td>
<td>767</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>#2 Energy Efficiency Cool loading (UCI)</td>
<td>767</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>#3 Airfoil Self-noise (UCI)</td>
<td>1503</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>#4 CCPP (UCI)</td>
<td>9568</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>#5 Housing (UCI)</td>
<td>506</td>
<td>13</td>
<td>8</td>
</tr>
<tr>
<td>#6 Yacht Hydrodynamics (UCI)</td>
<td>308</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>#7 Bodyfat (Weka)</td>
<td>252</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>#8 pwLinear (Weka)</td>
<td>200</td>
<td>14</td>
<td>7</td>
</tr>
<tr>
<td>#9 AutoPrice (Weka)</td>
<td>159</td>
<td>15</td>
<td>9</td>
</tr>
<tr>
<td>#10 Parkinson (UCI)</td>
<td>5873</td>
<td>19</td>
<td>12</td>
</tr>
<tr>
<td>#11 Concrete (UCI)</td>
<td>1030</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>#12 ERA (Weka)</td>
<td>1000</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>#13 Pollution (UCI)</td>
<td>60</td>
<td>15</td>
<td>9</td>
</tr>
<tr>
<td>#14 ESL (Weka)</td>
<td>488</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>#15 CPU (UCI)</td>
<td>209</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>#16 CO₂ production rate</td>
<td>10421</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>#17 Heavy Duty</td>
<td>10421</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>#18 Lean Loading</td>
<td>10421</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>#19 Absorption Efficiency</td>
<td>10421</td>
<td>8</td>
<td>6</td>
</tr>
</tbody>
</table>
As mentioned earlier, the primary evaluation criterion is fidelity of the generated rule sets to the originally trained ANN. While the accuracy of the PWL-ANN algorithm is also considered, the predictive accuracy of the PWL-ANN model is constrained by the predictive accuracy of the trained ANN model. In other words, if the trained ANN model has low accuracy, then the PWL-ANN model of this trained ANN model would also have low accuracy. To evaluate fidelity, the values of $R^2$ and mean square error (MSE) given by the ANN and the PWL-ANN models are compared. MSE measures the deviation of the predicted value to the actual value, and $R^2$ shows how close the actual values are to the fitted line. The values of $R^2$ and MSE can be calculated as follows:

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

where $y_i$ = the expected output of the $i^{th}$ dataset, $\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$ and $\hat{y}$ = predicted output.

$$MSE = \frac{\sum_i (y_i - \hat{y}_i)^2}{n}$$

Since an important research objective is to generate explicit information from the ANN models, the evaluation criterion adopted is that the PWL-ANN algorithm would produce similar values of $R^2$ and MSE as those produced by the ANN model on all the datasets. To more easily compare the results from the applications of the algorithm to the datasets, the percentage difference between the MSE given by the proposed PWL-ANN algorithm is compared to the MSE given by the trained ANN. The MSE percentage difference is given by:

$$MSE \% \text{ difference} = \text{Absolute} \left( \frac{MSE_{PWL-ANN} - MSE_{ANN}}{MSE_{ANN}} \right) \times 100\%$$
Table 3.2 and Table 3.3 summarize the MSE and $R^2$ given by the ANN and the PWL-ANN approaches, respectively, which show the accuracies of the ANN and PWL-ANN models.
Table 3.2 – MSE given by the ANN model and the PWL-ANN approaches

<table>
<thead>
<tr>
<th></th>
<th>ANN</th>
<th>PWL-ANN Approach 1</th>
<th>PWL-ANN Approach 2</th>
<th>PWL-ANN Approach 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>9.52E-01</td>
<td>4.51E+00</td>
<td>1.29E+00</td>
<td>1.10E+00</td>
</tr>
<tr>
<td>#2</td>
<td>3.54E+00</td>
<td>3.92E+00</td>
<td>3.54E+00</td>
<td>3.57E+00</td>
</tr>
<tr>
<td>#3</td>
<td>1.61E+01</td>
<td>1.66E+01</td>
<td>1.66E+01</td>
<td>1.65E+01</td>
</tr>
<tr>
<td>#4</td>
<td>2.47E+01</td>
<td>2.53E+01</td>
<td>2.53E+01</td>
<td>2.54E+01</td>
</tr>
<tr>
<td>#5</td>
<td>6.81E+00</td>
<td>7.72E+00</td>
<td>7.70E+00</td>
<td>7.67E+00</td>
</tr>
<tr>
<td>#6</td>
<td>1.49E+00</td>
<td>4.02E+00</td>
<td>3.96E+00</td>
<td>3.95E+00</td>
</tr>
<tr>
<td>#7</td>
<td>6.26E-01</td>
<td>7.49E-01</td>
<td>7.60E-01</td>
<td>7.61E-01</td>
</tr>
<tr>
<td>#8</td>
<td>1.40E+00</td>
<td>1.47E+00</td>
<td>1.46E+00</td>
<td>1.46E+00</td>
</tr>
<tr>
<td>#9</td>
<td>1.88E+06</td>
<td>2.21E+06</td>
<td>2.15E+06</td>
<td>2.19E+06</td>
</tr>
<tr>
<td>#10</td>
<td>1.93E+01</td>
<td>1.98E+01</td>
<td>1.98E+01</td>
<td>1.97E+01</td>
</tr>
<tr>
<td>#11</td>
<td>5.51E+01</td>
<td>5.84E+01</td>
<td>5.88E+01</td>
<td>5.85E+01</td>
</tr>
<tr>
<td>#12</td>
<td>3.56E+00</td>
<td>3.54E+00</td>
<td>3.60E+00</td>
<td>3.58E+00</td>
</tr>
<tr>
<td>#13</td>
<td>4.98E+02</td>
<td>5.98E+02</td>
<td>5.99E+02</td>
<td>5.94E+02</td>
</tr>
<tr>
<td>#14</td>
<td>4.59E-01</td>
<td>3.69E-01</td>
<td>3.67E-01</td>
<td>3.64E-01</td>
</tr>
<tr>
<td>#15</td>
<td>1.78E+01</td>
<td>7.09E+01</td>
<td>4.57E+01</td>
<td>4.78E+01</td>
</tr>
<tr>
<td>#16</td>
<td>1.24E-03</td>
<td>1.40E-03</td>
<td>1.40E-03</td>
<td>1.40E-03</td>
</tr>
<tr>
<td>#17</td>
<td>4.89E+07</td>
<td>5.13E+07</td>
<td>5.12E+07</td>
<td>5.02E+07</td>
</tr>
<tr>
<td>#18</td>
<td>3.04E-03</td>
<td>3.03E-03</td>
<td>3.03E-03</td>
<td>3.04E-03</td>
</tr>
<tr>
<td>#19</td>
<td>6.44E-03</td>
<td>6.73E-03</td>
<td>6.74E-03</td>
<td>6.72E-03</td>
</tr>
</tbody>
</table>
Table 3.3 – $R^2$ given by the ANN model and the PWL-ANN approaches

<table>
<thead>
<tr>
<th></th>
<th>ANN</th>
<th>PWL-ANN Approach 1</th>
<th>PWL-ANN Approach 2</th>
<th>PWL-ANN Approach 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>0.9907</td>
<td>0.9557</td>
<td>0.9874</td>
<td>0.9892</td>
</tr>
<tr>
<td>#2</td>
<td>0.9609</td>
<td>0.9566</td>
<td>0.9609</td>
<td>0.9605</td>
</tr>
<tr>
<td>#3</td>
<td>0.6618</td>
<td>0.6509</td>
<td>0.6208</td>
<td>0.6522</td>
</tr>
<tr>
<td>#4</td>
<td>0.9150</td>
<td>0.9130</td>
<td>0.9130</td>
<td>0.9129</td>
</tr>
<tr>
<td>#5</td>
<td>0.9193</td>
<td>0.9086</td>
<td>0.9088</td>
<td>0.9092</td>
</tr>
<tr>
<td>#6</td>
<td>0.9935</td>
<td>0.9825</td>
<td>0.9827</td>
<td>0.9828</td>
</tr>
<tr>
<td>#7</td>
<td>0.9910</td>
<td>0.9893</td>
<td>0.9891</td>
<td>0.9891</td>
</tr>
<tr>
<td>#8</td>
<td>0.9298</td>
<td>0.9261</td>
<td>0.9298</td>
<td>0.9266</td>
</tr>
<tr>
<td>#9</td>
<td>0.9454</td>
<td>0.9355</td>
<td>0.9373</td>
<td>0.9362</td>
</tr>
<tr>
<td>#10</td>
<td>0.8311</td>
<td>0.8271</td>
<td>0.8271</td>
<td>0.8277</td>
</tr>
<tr>
<td>#11</td>
<td>0.8023</td>
<td>0.7904</td>
<td>0.7893</td>
<td>0.7900</td>
</tr>
<tr>
<td>#12</td>
<td>0.0949</td>
<td>0.0981</td>
<td>0.0838</td>
<td>0.0875</td>
</tr>
<tr>
<td>#13</td>
<td>0.8691</td>
<td>0.8429</td>
<td>0.8426</td>
<td>0.8439</td>
</tr>
<tr>
<td>#14</td>
<td>0.7724</td>
<td>0.8172</td>
<td>0.8182</td>
<td>0.8195</td>
</tr>
<tr>
<td>#15</td>
<td>0.9993</td>
<td>0.9970</td>
<td>0.9981</td>
<td>0.9980</td>
</tr>
<tr>
<td>#16</td>
<td>0.9359</td>
<td>0.9276</td>
<td>0.9276</td>
<td>0.9276</td>
</tr>
<tr>
<td>#17</td>
<td>0.7541</td>
<td>0.7422</td>
<td>0.7426</td>
<td>0.7478</td>
</tr>
<tr>
<td>#18</td>
<td>0.7709</td>
<td>0.7716</td>
<td>0.7715</td>
<td>0.7709</td>
</tr>
<tr>
<td>#19</td>
<td>0.7804</td>
<td>0.7704</td>
<td>0.7703</td>
<td>0.7707</td>
</tr>
</tbody>
</table>
Table 3.4 summarizes the percentage differences between the MSE given by the trained ANN models and the PWL-ANN models and the results are plotted in Figure 3.3. It can be seen that the percentage differences are low: aside from dataset #1 (approach 1 and 2), #6, #7, #13 and #15, most of the datasets demonstrate reasonable fidelity to the original ANN with a percentage difference in MSE of below 20%. The low percentage differences between the two MSE’s demonstrate good fidelity of the PWL-ANN model to the original trained ANN.

When the percentage difference between the MSE’s is high, it means the PWL-ANN model demonstrates poor fidelity to the original ANN model. The analysis of the percentage difference between the MSE’s and R²’s has two objectives. First, the analysis aims to determine which of the three approaches listed earlier gives the best general performance. Secondly, it aims to investigate the causes for the poor performance of the algorithm on a particular dataset; the approach adopted is to analyze the performance of the three approaches on each dataset. The datasets identified with high percentage differences between the MSE’s and R² will be the focus of the analysis, and the analysis is presented as follows.
Table 3.4 – MSE percentage difference between ANN and the three PWL-ANN approaches

<table>
<thead>
<tr>
<th></th>
<th>PWL-ANN Approach 1</th>
<th>PWL-ANN Approach 2</th>
<th>PWL-ANN Approach 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>373.75%</td>
<td>35.25%</td>
<td>15.52%</td>
</tr>
<tr>
<td>#2</td>
<td>10.94%</td>
<td>0.00%</td>
<td>0.81%</td>
</tr>
<tr>
<td>#3</td>
<td>3.20%</td>
<td>3.16%</td>
<td>2.83%</td>
</tr>
<tr>
<td>#4</td>
<td>2.43%</td>
<td>2.40%</td>
<td>2.56%</td>
</tr>
<tr>
<td>#5</td>
<td>13.31%</td>
<td>13.06%</td>
<td>12.61%</td>
</tr>
<tr>
<td>#6</td>
<td>169.59%</td>
<td>165.72%</td>
<td>164.96%</td>
</tr>
<tr>
<td>#7</td>
<td>19.75%</td>
<td>21.53%</td>
<td>21.66%</td>
</tr>
<tr>
<td>#8</td>
<td>5.25%</td>
<td>4.05%</td>
<td>4.52%</td>
</tr>
<tr>
<td>#9</td>
<td>17.99%</td>
<td>14.75%</td>
<td>16.76%</td>
</tr>
<tr>
<td>#10</td>
<td>2.37%</td>
<td>2.37%</td>
<td>2.00%</td>
</tr>
<tr>
<td>#11</td>
<td>6.04%</td>
<td>6.59%</td>
<td>6.21%</td>
</tr>
<tr>
<td>#12</td>
<td>-0.35%</td>
<td>1.23%</td>
<td>0.82%</td>
</tr>
<tr>
<td>#13</td>
<td>20.02%</td>
<td>20.28%</td>
<td>19.25%</td>
</tr>
<tr>
<td>#14</td>
<td>-19.65%</td>
<td>-20.11%</td>
<td>-20.67%</td>
</tr>
<tr>
<td>#15</td>
<td>298.94%</td>
<td>156.78%</td>
<td>168.77%</td>
</tr>
<tr>
<td>#16</td>
<td>13.01%</td>
<td>12.99%</td>
<td>12.99%</td>
</tr>
<tr>
<td>#17</td>
<td>4.84%</td>
<td>4.67%</td>
<td>2.57%</td>
</tr>
<tr>
<td>#18</td>
<td>-0.27%</td>
<td>-0.24%</td>
<td>0.02%</td>
</tr>
<tr>
<td>#19</td>
<td>4.56%</td>
<td>4.58%</td>
<td>4.42%</td>
</tr>
</tbody>
</table>
3.2.1 Overall Performance of the Three Approaches

In both approaches 2 and 3, a small value of d is used, which means more data points are tested as the potential break points. Using these approaches ensure a more fine-grained approximation of the curve is implemented. Hence, the performances of Approach 2 and 3 can be expected to be better than Approach 1. This hypothesis is valid and it can be seen in Table 3.5 that approach 1 gives the best fidelity for only 4 datasets, while approaches 2 and 3 give the best performance for 5 and 9 datasets, respectively. The difference between approach 2 and 3 is that approach 2 uses half of the values on the sigmoid curve, whereas approach 3 uses the absolute values and all the data points on the sigmoid curve. Since approach 3 outperforms approach 2, we can conclude that among the three approaches, the best performance is given when the full range of values is included and when a finer grained approximation is used. Thus, in future work, the absolute values and a small value of d will be used for the PWL approximation.
Table 3.5 – Performances of the PWL-ANN models

<table>
<thead>
<tr>
<th>Approach</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of dataset with the best performance</td>
<td>4</td>
<td>5</td>
<td>9</td>
</tr>
</tbody>
</table>

Note: Data #16 is a draw for both Approach 2 and 3.
3.2.2 Performance of the Three Approaches on Each Individual Dataset

An analysis of the R$^2$ values in Table 3.3 reveals that the PWL-ANN models give close performances to the ANN models on all datasets, even for the dataset on which the ANN model did not give good predictive performance. Based on the premise that a R$^2$ value that is close to 1 indicates higher predictive accuracy of the model, it can be seen from Table 3.3 that the R$^2$ given by dataset #12 is only 0.0949, which reflects low accuracy, and the values given by the PWL-ANN models are 0.0981, 0.0838 and 0.0875, which show similar performance to the ANN model.

Table 3.4 and Figure 3.3 show that aside from dataset #1 (approach 1 and 2), #6, #7, #13 and #15, most of the datasets demonstrate reasonable fidelity to the original ANN with a percentage difference in MSE of below 20%. The analysis aims to identify the causes for the poor performances of the PWL-ANN algorithms. The datasets with a MSE percentage difference greater than 150% include (aside from dataset #1) datasets #6 and #15, as shown in italics in Table 3.4 and indicated by the tall columns in Figure 3.3. These datasets deserve careful examination because of their high MSE percentage differences. Dataset #1 will not be examined because its performance was improved from 373.75% in approach 1 to 15.52% in approach 3. Similarly, datasets #7 and #13 will not be examined because their MSE percentage differences are just over 20%, which can be considered acceptable. Thus, the datasets to be examined include only datasets #6 and #15.
In some of the cases, the PWL-ANN models even outperformed the ANN models in terms of accuracies: in dataset #14, the PWL-ANN models give higher $R^2$'s than the ANN model. As pointed out by Zhou (2004), the criteria of accuracy and fidelity can be contradictory. If the assumption is that the underlying ANN model gives good accuracies, then if the proposed rule extraction algorithm has a high fidelity to the ANN model, the rule extraction algorithm will also demonstrate good accuracy. This will fulfill both of the objectives on fidelity and accuracy. However, if there is a dataset for which the underlying ANN model does not predict well, then the rule extraction algorithm can only accomplish one of the two objectives: either it models the ANN model very well, such that the algorithm also gives poor prediction, or it generates a set of rules that gives good accuracy; and the result of data #14 demonstrates this contradiction. Since our primary evaluation criterion is fidelity, it is not expected that the PWL-ANN model should outperform the ANN model. Therefore, the PWL-ANN modeling result of dataset #14 is considered not satisfactory.
Figure 3.3 – MSE percentage difference between the ANN models and the PWL-ANN models
3.3 Two Hypotheses to Explain Poor Performance of the PWL-ANN Algorithm

3.3.1 Hypothesis 1

*If the PWL approximation of all nodes is highly accurate (with $R^2 > 0.99$), then the fidelity of the PWL-ANN model should be high.*

This hypothesis suggests a causal relationship between the accuracy of the PWL approximation as indicated by the $R^2$ and MSE values and the fidelity of the PWL-ANN model to the original ANN model. *Table 3.6* and *Table 3.7* illustrate this relationship for datasets #6 and #15. The two tables show the values of MSE and $R^2$ of each node given by the PWL equations in the three approaches for datasets #6 and #15, respectively. It is assumed that if the PWL approximation is highly accurate as indicated by a $R^2$ value of greater than 0.99, then the PWL-ANN model generated by the algorithm will demonstrate high fidelity to the original ANN, and this will be indicated by a low percentage difference value. This hypothesis was supported by dataset #6 but not by dataset #15. By analyzing the $R^2$ values in Table 3.6, it can be seen that some of the nodes in dataset #6 have an $R^2$ of less than 0.99 (the values are indicated in italics). In this case, the PWL-ANN models demonstrate poor fidelity to the original ANN models. By contrast, in Table 3.7, approaches 2 and 3 both gave $R^2$ values that are greater than 0.99 for all the nodes in dataset #15, and yet the MSE percentage differences are still above 150%. This shows that high fidelity of the PWL-ANN model to the original ANN model cannot be guaranteed even when each individual node has been accurately approximated.
Table 3.6 – Dataset #6: Comparison of accuracies of the PWL approximation for each node with fidelity of the PWL-ANN model to the original ANN model

<table>
<thead>
<tr>
<th>Node 1</th>
<th>Node 2</th>
<th>Node 3</th>
<th>Node 4</th>
<th>Node 5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Approach 1</strong> % difference = 169.59%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>MSE</strong></td>
<td>2.58E-04</td>
<td>9.20E-06</td>
<td>5.30E-06</td>
<td>1.56E-04</td>
</tr>
<tr>
<td><strong>R²</strong></td>
<td>0.9878</td>
<td>0.9974</td>
<td>0.997</td>
<td>0.9768</td>
</tr>
<tr>
<td><strong>Approach 2</strong> % difference = 165.72%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>MSE</strong></td>
<td>2.54E-04</td>
<td>9.20E-06</td>
<td>5.30E-06</td>
<td>1.55E-04</td>
</tr>
<tr>
<td><strong>R²</strong></td>
<td>0.988</td>
<td>0.9974</td>
<td>0.997</td>
<td>0.9769</td>
</tr>
<tr>
<td><strong>Approach 3</strong> % difference = 164.96%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>MSE</strong></td>
<td>2.54E-04</td>
<td>9.20E-06</td>
<td>5.30E-06</td>
<td>1.55E-04</td>
</tr>
<tr>
<td><strong>R²</strong></td>
<td>0.988</td>
<td>0.9974</td>
<td>0.997</td>
<td>0.9769</td>
</tr>
</tbody>
</table>

Table 3.7 – Data #15: Comparison of accuracies of the PWL approximation for each node with fidelity of the PWL-ANN model to the original ANN model

<table>
<thead>
<tr>
<th>Node 1</th>
<th>Node 2</th>
<th>Node 3</th>
<th>Node 4</th>
<th>Node 5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Approach 1</strong> % difference = 298.94%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>MSE</strong></td>
<td>2.25E-05</td>
<td>3.00E-07</td>
<td>8.90E-06</td>
<td>6.34E-05</td>
</tr>
<tr>
<td><strong>R²</strong></td>
<td>0.9986</td>
<td>0.9997</td>
<td>0.9993</td>
<td>0.988</td>
</tr>
<tr>
<td><strong>Approach 2</strong> % difference = 156.78%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>MSE</strong></td>
<td>2.16E-05</td>
<td>3.00E-07</td>
<td>8.60E-06</td>
<td>4.45E-05</td>
</tr>
<tr>
<td><strong>R²</strong></td>
<td>0.9986</td>
<td>0.9997</td>
<td>0.9994</td>
<td>0.9916</td>
</tr>
<tr>
<td><strong>Approach 3</strong> % difference = 168.77%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>MSE</strong></td>
<td>2.16E-05</td>
<td>3.00E-07</td>
<td>8.60E-06</td>
<td>4.73E-05</td>
</tr>
<tr>
<td><strong>R²</strong></td>
<td>0.9986</td>
<td>0.9997</td>
<td>0.9994</td>
<td>0.991</td>
</tr>
</tbody>
</table>
3.3.2 Hypothesis 2

*Insufficiency of the three-piece linear equation approximation*

Since it has been established from the analysis of hypothesis 1 that accurate approximation of each node of the original ANN model cannot guarantee good fidelity of the PWL-ANN model to the original ANN model, the analysis now turns to investigate more specific values within the nodes. The second hypothesis assumes that the poor performance of the PWL-ANN algorithm is due to the insufficiency of the three-piece linear equation in approximating the curve in the sigmoid functions. In other words, while some of the data values that fall on the curve of the sigmoid functions in some of the hidden nodes have significant influences on the overall performance of the PWL-ANN, these points on the curve are not adequately modeled. As a consequence, the PWL-ANN model shows poor fidelity to the original ANN model. It is important to note that this hypothesis does not assume that omission of all the points that lie in the curves of all the hidden nodes cause the poor fidelity to the original ANN model. Rather, the omission of some combination of points in some nodes is the culprit. If this hypothesis is true, a more fine-grained PWL approximation, such as a five-piece linear approximation, on the appropriate hidden nodes would improve the performance of the algorithm.

To test the hypothesis on the insufficiency of the three-piece linear equations, the data points that lie on the curve of the hidden nodes are removed one at a time, and these subsets of data are used to generate new sets of 3-piece linear equations for the PWL-
ANN models. The data points are eliminated one hidden node at a time so as to reveal which sets of data points have positive effects on the MSE of the PWL-ANN model. Since application of the PWL-ANN algorithm to dataset #6 demonstrated poor performance, this elimination process was applied to dataset #6. The results are shown in Table 3.8, whose left column indicates the modeled dataset, and the right column indicates the MSE values given by the subsets. The first row of Table 3.8 shows the MSE values derived from applying the PWL-ANN algorithm to the original dataset using Approach 3. Subset 1 refers to the dataset that has the data points lying on the curve of the first hidden node removed, and Subset 2 refers to the data points lying on the curve of the second node removed and so on.

It can be seen from Table 3.8 that by removing the data points that lie on the curves of nodes 1 and 5, the MSE’s of the PWL-ANN models have decreased and improved from that given by the original model. While removing data points from node 2 and 4 increased the MSE values from 3.95 to 3.96 and 4.18, respectively. Based on these results, we generate a new subset, called subset B, which is produced by removing data points that lie on the curves of node 1 and 5, because these nodes caused an improvement on the MSE value. The MSE given by applying the PWL-ANN algorithm to subset B is significantly lower than that given by approach 3, as shown in Table 3.9. The percentage difference between the MSE given by the PWL-ANN model and the original ANN model also decreased from 164.96% to 42.42%.
Table 3.8 – MSE’s given by PWL-ANN modeling for original dataset #6 and its subset

<table>
<thead>
<tr>
<th></th>
<th>MSE</th>
<th>% difference with MSE from Approach 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approach 3</td>
<td>3.95</td>
<td>--</td>
</tr>
<tr>
<td>Subset 1</td>
<td>2.62</td>
<td>-33.59%</td>
</tr>
<tr>
<td>Subset 2</td>
<td>3.96</td>
<td>0.29%</td>
</tr>
<tr>
<td>Subset 3</td>
<td>3.95</td>
<td>0.00%</td>
</tr>
<tr>
<td>Subset 4</td>
<td>4.18</td>
<td>5.79%</td>
</tr>
<tr>
<td>Subset 5</td>
<td>3.21</td>
<td>-18.80%</td>
</tr>
</tbody>
</table>

Table 3.9 – Performance of PWL-ANN modeling on subset B of dataset #6 using approach 3

<table>
<thead>
<tr>
<th></th>
<th>PWL-ANN MSE</th>
<th>MSE % diff. with ANN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approach 3</td>
<td>3.95</td>
<td>164.96%</td>
</tr>
<tr>
<td>Subset B</td>
<td>2.08</td>
<td>42.42%</td>
</tr>
</tbody>
</table>
The result from this analysis suggests hypothesis 2 is valid. It shows that the accuracy and fidelity of the PWL-ANN model can be improved by better approximating the sigmoid function. Hence, future work involves increasing the “pieces” in the piece-wise linear equation from 3-piece to 5-piece, so as to improve the approximation of the curves in the hidden nodes.

3.4 Discussion

The analysis results suggested that the accuracy of the PWL-ANN model can be improved by increasing the number of pieces, or breakpoints (BPs) in the piece-wise linear equations. In fact, Setiono et al. (2002) suggested a 5-piece approximation algorithm in their work and their results showed that the 5-piece approximation outperformed the 3-piece approximation. Some weaknesses in their work include: their application dataset has less than 700 data points, and the complexity of the 5-piece linear application algorithm was not considered. In fact, a 5-piece linear approximation requires significantly more computation time, and this is especially true for industrial datasets, e.g. the CO2 capture plant datasets in our study have over 10,000 data points. A promising step forward in this research is to extend the brute-force search algorithm explained in Section 3.1 to locate more break points. Instead of locating only one breakpoint for a 3-piece approximation because of symmetry of the sigmoid function, a 5-piece linear approximation will require locating more than one breakpoint. It will also be necessary to examine and consider the computing time required to process a 3-piece (1 breakpoint) and a 5-piece (2 breakpoints) approximation on different sizes of datasets.
While increasing the number of breakpoints can improve the accuracy of the PWL-ANN model, this approach has the possible trade-off of increasing complexity of the overall system. By increasing the number of pieces or linear equations at each node, the possible combinations of equations also increases significantly. With a 3-piece approximation of each hidden node, the number of possible combinations of equations is $3^j$, with $j$ being the number of hidden nodes. If all hidden nodes are approximated with 5-pieces, the number of possible combinations of equations will become $5^j$. This trade-off between the number of pieces and complexity of the system needs to be considered. Therefore, the number of pieces within the piecewise equation per node should be determined based on consideration of both system complexity and desired accuracy of the PWL-ANN model.

### 3.5 Conclusion

In this chapter, we presented an artificial neural network rule extraction algorithm, called PWL-ANN, based on a 3-piece linear approximation of the hidden neuron activation function, which is assumed to be a sigmoid function. By replacing the activation functions with the PWL equations, rules in the form of linear equations that relate the inputs and outputs of a given dataset can be generated. Three approaches for solving the PWL approximations have been presented and all three approaches were tested on 19 datasets. The results show that the best performance is given when the full range of data values is included and when a finer grained approximation is used. The algorithm gives satisfactory fidelity on sixteen out of the nineteen tested datasets. An analysis of the values of $R^2$ given by the PWL approximation on the hidden neurons and the overall
output shows that in addition to the precision of the approximation of each node, other factors also play a role in determining the fidelity of the PWL-ANN algorithm. Further investigation is required to determine what those factors may be.
CHAPTER 4 ENHANCED METHODOLOGY – THE ENHANCED-PWL-ANN ALGORITHM

This chapter presents a decomposition artificial neural network (ANN) rule extraction algorithm, which is an extension of the PWL-ANN (Piecewise Linear ANN). Based on the analysis in Chapter 3, the PWL-ANN algorithm needs to be further improved in terms of accuracy and fidelity of the generated models as well as comprehensibility of the extracted rules. Therefore, an enhanced version of the PWL-ANN algorithm (Chan & Chan, 2017) which demonstrates higher accuracy, fidelity and comprehensibility of the extracted rules was developed. The objectives of this chapter are as follows:

1. To present the developed enhanced-PWL-ANN algorithm that can generate more accurate PWL-ANN models compared to those produced by the preliminary PWL-ANN algorithm in the previous chapter. The performance of the enhanced-PWL-ANN algorithm is verified using datasets from the UCI repository (Lichman, 2013);

2. To further simplify the rule set by applying the decision tree algorithm to the extracted rule sets given by the enhanced-PWL-ANN algorithm.

3. To illustrate performance of both the enhanced-PWL-ANN algorithm, and application of the decision tree algorithm for simplification of the extracted rules. Some datasets from UCI will be used.

The enhanced-PWL-ANN algorithm follows the scope of work that is covered by the PWL-ANN algorithm, with the exception of the expressive power of the rules. The scope of work of the enhanced-PWL-ANN algorithm is as follows:
1. The originally trained ANN models are assumed to be a feedforward backpropagation neural network models.

2. The originally trained ANN models are assumed to have one hidden layer with either sigmoid or hyperbolic function as the hidden activation function and the linear function as the output activation function.

3. The target problems are assumed to be regression problems related to engineering domains.

4. The expressive power of the generated rules in Chapter 3 is in terms of linear numeric functions in the form of

\[ y = a_1 x_1 + a_2 x_2 + \ldots + a_n x_n + b \]

In this study, the rules generated by the enhanced-PWL-ANN algorithm are in the form of IF-THEN rules.

The PWL-ANN algorithm proposed in Chapter 3 will be called PWL-ANN and the enhanced version proposed in this chapter will be called enhanced-PWL-ANN algorithm in the rest of the chapter.

4.1 Methodology

Similar to the PWL-ANN algorithm, the enhanced-PWL-ANN algorithm involves three main steps:

1. Training the neural network for a given dataset, the ANN model obtained is called the originally trained ANN model.
2. Approximating the activation functions of the hidden neurons of the originally trained ANN model and generate the PWL-ANN models.

3. Extracting rules from the generated PWL-ANN models

While Steps 1 and 3 remain unchanged from the PWL-ANN algorithm presented in Chapter 3, some modifications are made to step 2 in the enhanced version, which aimed to improve accuracy and fidelity of the enhanced PWL-ANN models to the originally trained ANN model. At the same time, the enhanced PWL-ANN algorithm aimed to reduce computation time and increase comprehensibility of the extracted rules. For implementation, Rapidminer (trademark of Rapidminer) was used to train the original neural network model from a given dataset, and the enhanced-PWL-ANN algorithm was written in R.

4.1.1 Training the neural network for a given dataset

The originally trained ANN models were generated using Rapidminer’s neural network model (trademark of Rapidminer) and the parameter settings used are as follows:

- Normalization: [-1, 1]
- Number of hidden layer: 1
- Number of nodes per hidden layer: \((\text{number of input attributes} + \text{number of output attributes})/2 +1\)
- Hidden layer activation function: sigmoid function
- Output layer activation function: linear function
- Number of training cycle: 500
- Learning Rate: 0.3
• Momentum: 0.2
• Validation: 20-fold cross-validation

Rapidminer evaluates the performance of the trained ANN model by the value of root mean squared error. It also provides information on the trained ANN model including the weights of the nodes of the originally trained ANN model which is used in the PWL approximation algorithm.

4.1.2 Approximating the activation functions of the hidden neurons of the trained ANN model and generating the enhanced-PWL-ANN models.

Three modifications were made in the enhanced-PWL-ANN algorithm from the original PWL-ANN presented in Chapter 3:

1. The brute force search is extended to locate N number of breakpoints and therefore the hidden activation functions can be approximated to (N+1)-piece linear equations;

2. A sampling algorithm is included as an option to manage computation time of the algorithm (Please refer to Appendix C for details on the evaluation of computation time versus data size, and the performance of the sampling algorithm);

3. A sensitivity analysis (SA) algorithm is implemented to decide the number of breakpoints used in each of the hidden nodes of the originally trained ANN model.

The inputs required for the algorithm include:
(1) The values of the predictor and predicted attributes,

(2) The calculated values of the predicted attributes given by the ANN model,

(3) The weights of the hidden neurons,

(4) The weights of the output neurons,

(5) The activation function used by the trained ANN model (either sigmoid or hyperbolic function),

(6) Minimum distance between breakpoints ($l$),

(7) Maximum number of breakpoints (maxBP),

(8) If sampling algorithm was used (either TRUE or FALSE)

(9) If sampling algorithm was used, the sampling size ($D_{sampling}$), and

(10) The allowable error of the generated PWL-ANN model in terms of MSE percentage difference (E). The MSE percentage difference is given by:

$$E = \text{Absolute}(\frac{MSE_{PWL-ANN} - MSE_{ANN}}{MSE_{ANN}}) \times 100\%$$

The steps involved in approximating the hidden activation functions of a trained ANN model are as follows:

1. The weighted input of each node is calculated. The weighted input, $\xi_j$, of node $j$ is given by

$$\xi_j = \sum_{i=1}^{n} w_{ji} x_i + \theta_j$$

Where $x_i$ is the value of input $i$,

$w_{ji}$ is the weight of input $i$ to node $j$, and

$\theta_j$ is the bias value of node $j$. 
2. The sigmoid (or hyperbolic tangent, depending on the trained ANN model) values of each node are calculated. The sigmoid values are given by:

\[ f(\xi) = 1/1 + e^{-\xi} \]

and the hyperbolic tangent values are given by:

\[ f(\xi) = (e^\xi - e^{-\xi})/(e^\xi + e^{-\xi}) \]

3. If the choice of sampling is selected, then the sampling algorithm is applied, and a sample dataset will be generated based on the given dataset and desired sample data size; the details of the algorithm are presented in Section 4.1.2.1.

4. Each of the nodes is approximated to the user-specified maximum break-point using brute-force search. The user-specified maximum break-point, and is called maxBP; the details of the algorithm is presented in Section 4.1.2.2.

5. The outputs of the enhanced-PWL-ANN algorithm are first calculated using only 2 breakpoints for all nodes. The base case of using two breakpoints (BPs) for all hidden nodes is adopted because two is the lowest number of BP that can be used for each node and it gives the least number of rules. If the MSE error given when 2 breakpoints were used satisfied E, then the enhanced-PWL-ANN model with 2 breakpoints is returned as the solution.

6. However, if the base case of identifying only 2 breakpoints does not give satisfactory results, then the sensitivity analysis will be conducted and a higher number of BP will be determined. At any point in this process of testing different cases, if the MSE percentage difference between the output of the originally trained ANN model and the calculated enhanced-PWL-ANN output is less than the user-specified error rate of E, then the set of PWL equations for that case will
be returned; the details of the sensitivity analysis algorithm is presented in Section 4.1.2.3.

If the SA algorithm fails to find a solution, then no solution can be found with the specified maxBP and E. The algorithm will return the set of equations with the lowest MSE percentage difference. However, the user can also consider increasing the number of breakpoints or lowering the error rate, and restarting the testing process.

The pseudo code of the steps involved is shown in Figure 4.1, and the flow chart of the algorithm can be found in Figure 4.2.
Figure 4.1 – Pseudo code of the enhanced-PWL-ANN algorithm
Figure 4.2 – Flow chart of the enhanced-PWL-ANN algorithm
4.1.2.1 Sampling algorithm

The sampling algorithm generates a subset of data by taking every $x$ point in the data set. The value of $x$ is given by dividing the actual data size, $D$, by the desired size of subset, $D_{\text{sample}}$, which is user-determined. The sampling algorithm is introduced as an option to manage computation time of the algorithm. The user can also choose to completely omit the sampling algorithm.

4.1.2.2 Extended brute-force search algorithm

The brute-force search algorithm employed to locate the breakpoints of the PWL equations is based on the dynamic programming algorithm suggested by Bai & Perron (2003). Bai & Perron (2003) proposed to first construct a matrix of sums of squared residuals (SSR), then a search is performed through the SSR matrix for the best locations of the breakpoints (BPs) with the minimum total SSR. The sum of squared residuals measures the deviation of the fitted values to the actual values; and it is given by

$$SSR = \sum_i (y_i - \hat{y}_i)^2$$

where $y_i =$ the expected output of the $i^{th}$ dataset, and $\hat{y} =$ predicted output.

MSE is given by SSR divided by the total number of data points. Assuming there are $M$ number of data points in a given dataset, the SSR matrix will be a triangular matrix of size $M \times M$. Note that $M$ is equal to $D_{\text{sample}}$ if the sampling algorithm is used, otherwise $M$ is equal to $D$. The entries of the matrix consists of the estimated sum of squared residuals of all possible pieces for the specified number of breakpoints (maxBP) and minimum distance between breakpoints ($l$). An example of the SSR matrix can be seen in
Table 4.1. In this example, the data size, M, is 20, the maximum number of breakpoints, maxBP, is 2, the minimum distance between the BPs, \( l \), is 3. The minimum length of each piece is the minimum distance between the breakpoints (BPs) plus the two end BPs, therefore, it is 5 in this example. The row index represents the start of a piece and the column index represents the end of the piece. For example, the value in row 1 and column 5, which is represented as SSRMatrix[1, 5], is the SSR of the piece starting from data point 1 and ends at data point 5 and data point 5 is a breakpoint. As indicated in Table 4.1, not all possible pieces will be considered. A piece will only be considered as acceptable if it satisfies the following three conditions:

1. The length of the piece is at least \( l + 2 \).

The length of a piece has to satisfy the minimum distance between the breakpoints specified by the user. The minimum length of each piece is the minimum distance between the BPs plus the two end points. Therefore, a piece is only considered acceptable if it has a length of at least \( l+2 \). In Table 4.1, the pieces that violate this condition are indicated by “a”.

2. There is enough room for another piece of length at least \( l + 2 \) in front of the piece.

This condition ensures that the piece before a chosen BP satisfies the minimum length requirement. In Table 4.1, the pieces that violate this condition are indicated by “c”. In this example, the piece starting from point 2 and ends at point 6 is not valid because the piece prior to it, i.e. the piece that starts from 1 and ends at 2, is too short. In fact, any piece that starts with data point 2, 3 and 4 is invalid in the example shown in Table 4.1.

3. There is enough room for maxBP pieces with length of at least \( l + 2 \).
This condition is to ensure there is enough room for all the pieces with the minimum length to fit into the dataset. The pieces that violate this condition are indicated by “b” in Table 4.1. For example, the piece starting with point 1 and ends with point 12 is invalid because it is too long and there is not enough room left to fit two more pieces with minimum length of 5 into it.

Only the pieces indicated by “*” in Table 4.1 are considered as valid pieces.
Table 4.1 – Example of a sums of squared residuals (SSR) matrix: \( M = 20, \ maxBP = 2, \ l = 3 \)

\[
\begin{array}{cccccccccccccccc}
1 & a & a & a & a & * & * & * & * & * & b & b & b & b & b & b & b & b & b & b \\
2 & a & a & a & a & c & c & c & c & c & c & c & c & c & c & c & c & c & c & c \\
3 & a & a & a & a & c & c & c & c & c & c & c & c & c & c & c & c & c & c & c \\
4 & a & a & a & a & c & c & c & c & c & c & c & c & c & c & c & c & c & c & c \\
5 & a & a & a & a & * & * & * & * & * & * & * & * & * & b & b & b & b & b & b \\
6 & a & a & a & a & * & * & * & * & * & * & * & * & * & b & b & b & b & b & b \\
7 & a & a & a & a & * & * & * & * & * & * & * & * & * & * & * & * & * & * & * \\
8 & a & a & a & a & * & * & * & * & * & * & * & * & * & * & * & * & * & * & * \\
9 & a & a & a & a & * & * & * & * & * & * & * & * & * & * & * & * & * & * & * \\
10 & a & a & a & a & * & * & * & * & * & * & * & * & * & * & * & * & * & * & * \\
11 & a & a & a & a & * & * & * & * & * & * & * & * & * & * & * & * & * & * & * \\
12 & a & a & a & a & * & * & * & * & * & * & * & * & * & * & * & * & * & * & * \\
13 & a & a & a & a & * & * & * & * & * & * & * & * & * & * & * & * & * & * & * \\
14 & a & a & a & a & * & * & * & * & * & * & * & * & * & * & * & * & * & * & * \\
15 & a & a & a & a & * & * & * & * & * & * & * & * & * & * & * & * & * & * & * \\
16 & a & a & a & a & * & * & * & * & * & * & * & * & * & * & * & * & * & * & * \\
17 & a & a & a & a & * & * & * & * & * & * & * & * & * & * & * & * & * & * & * \\
18 & a & a & a & a & * & * & * & * & * & * & * & * & * & * & * & * & * & * & * \\
19 & a & a & a & a & * & * & * & * & * & * & * & * & * & * & * & * & * & * & * \\
20 & a & a & a & a & * & * & * & * & * & * & * & * & * & * & * & * & * & * & * \\
\end{array}
\]

Note:

- This piece will not be considered, since it will not satisfy minimum length of 5
- This piece will not be considered, since there will be no room for 2 pieces of length 5
- This piece will not be considered, since there will be no place for a piece with length 5 before it
- this is an allowable piece.
After the SSR matrix is generated, a search is performed on the SSR matrix for the combination of breakpoints with the lowest SSR. Let us consider the example shown in Table 4.1, where $M = 20$, $\text{maxBP} = 2$ and $l = 3$. The search begins with data point 1 and the first allowable breakpoint is at data point 5. Then the algorithm will search for the second breakpoint between data points 5 to 20 by considering all the possible points between 9 to 16 as the potential second breakpoint, and the search returns the one that gives the minimum SSR. In this case, the algorithm calculates all the following:

- Second breakpoint at 9: $\text{SSRMatrix}[5, 9] + \text{SSRMatrix}[9, 20]$
- Second breakpoint at 10: $\text{SSRMatrix}[5, 10] + \text{SSRMatrix}[10, 20]$
- Second breakpoint at 11: $\text{SSRMatrix}[5, 11] + \text{SSRMatrix}[11, 20]$
- Second breakpoint at 12: $\text{SSRMatrix}[5, 12] + \text{SSRMatrix}[12, 20]$
- Second breakpoint at 13: $\text{SSRMatrix}[5, 13] + \text{SSRMatrix}[13, 20]$
- Second breakpoint at 14: $\text{SSRMatrix}[5, 14] + \text{SSRMatrix}[14, 20]$
- Second breakpoint at 15: $\text{SSRMatrix}[5, 15] + \text{SSRMatrix}[15, 20]$
- Second breakpoint at 16: $\text{SSRMatrix}[5, 16] + \text{SSRMatrix}[16, 20]$

and returns the combination with the minimum value of total SSR. Assuming the minimum SSR is found when the second breakpoint is at 9, then, the first set of breakpoints becomes $(5, 9)$ and the total SSR is given by $\text{SSRMatrix}[1, 5] + \text{SSRMatrix}[5, 9] + \text{SSRMatrix}[9, 20]$. The search will move onto the next allowable first breakpoint, i.e. data point 6, and it will locate the second breakpoint between 6 to 20 by considering all the possible points as BP and returns the one that gives the lowest SSR. If this new set of BPs has a total SSR lower than the previous total SSR, then the new set of BPs becomes the optimal solution. The search will then move onto the next allowable first breakpoint and find the total SSR, which is compared with the previous minimum SSR found and so on, until all allowable pieces are considered and identified.
The construction of the SSR matrix is the most time consuming process in the enhanced-PWL-ANN algorithm. The sampling algorithm described in section 4.1.2.1 was introduced to manage the computation time involved.

4.1.2.3 Sensitivity analysis algorithm

There are two reasons why the sensitivity analysis (SA) algorithm is included in the algorithm:

1. While examining the sufficiency of the 3-piece equations in Chapter 3, it was discovered that not all hidden nodes in the originally trained ANN model have the same effect on the overall performance of the PWL-ANN model. Some hidden nodes contributed more than others towards the accuracy of the output, and these significant hidden nodes need to be approximated more accurately. Therefore, a sensitivity analysis algorithm that can select the significant hidden nodes, for which the number of breakpoints will be increased is designed and included in the enhanced PWL-ANN algorithm. This is done to achieve the desired higher accuracy.

2. As pointed out in Chapter 3, there is a trade-off between the number of breakpoints and the comprehensibility of the extracted rules. By increasing the number of PWL equations per hidden node of the originally trained ANN model, the potential number of rules increases significantly. Assuming there are j number of nodes, by increasing the PWL equations for each node from 3-piece to 5-piece, the possible number of multiple linear equations for the model increases from $3^j$
Therefore, it is important to only increase the number of breakpoints for the least number of hidden nodes in an effort to achieve the desired error rate. The SA algorithm will determine for which hidden node the number of BPs need to be increased, and how many pieces of linear segments are made per node, so as to achieve higher predictive accuracy.

Sensitivity analysis is defined as an investigation into how much the output of a model is affected by the changes of some or a combination of some input parameters (Sensitivity Analysis, 2016). In this case, the number of breakpoints per node is the varying parameter and the accuracy of the enhanced-PWL-ANN model is the affected output. Sensitivity analysis is conducted by varying the number of linear pieces per hidden node in the originally trained ANN model one node at a time and the change in the output of the enhanced-PWL-ANN model can be observed in an effort to find the optimal solution.

The steps involved in the sensitivity analysis algorithm are as follows (step 6 in pseudo code):

1. Increase the number of BP one hidden node at a time, while keeping all the other nodes at 2 BP, and calculate the enhanced-PWL-ANN model output and corresponding MSE. Record the number of BP per node and the MSE values in the SA matrix as shown in Table 4.2. In the example shown in Table 4.2, the originally trained ANN model consists of 5 hidden nodes. The MSE given by the base case model, i.e. when all nodes has 2 BPs, is 3.9874; and the table shows the MSE values given by the enhanced-PWL-ANN model by increasing the BP one node at a time while keeping the other nodes at 2BPs. For example, the value shown in the row for 4
BPs and column for node 1 is the MSE value of 2.1240 given by the model when node 1 has 4 BPs, while all the other nodes have 2 BP. Similarly, the value in the row for 6 BPs and column for node 1 is the MSE value of 1.8460 given by the model when node 1 has 6 BPs, while all the others have 2 BPs.

2. Check the MSE values given by the SA matrix. Since the goal is to only increase the number of breakpoints for the least number of hidden nodes, if any of the results of the MSE values from the SA satisfies E, the set of PWL equations that give the lowest E will be returned as the final result. This means a satisfactory model is found by increasing the breakpoints to either 4 or 6 only in the nodes identified in the SA matrix.
### Table 4.2 – Example of Sensitivity Analysis Matrix

<table>
<thead>
<tr>
<th></th>
<th>node1</th>
<th>node2</th>
<th>node3</th>
<th>node4</th>
<th>node5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 BP</td>
<td>3.9874</td>
<td>3.9874</td>
<td>3.9874</td>
<td>3.9874</td>
<td>3.9874</td>
</tr>
<tr>
<td>4 BP</td>
<td>2.1240</td>
<td>3.9884</td>
<td>3.9874</td>
<td>3.6483</td>
<td>3.9878</td>
</tr>
<tr>
<td>6 BP</td>
<td>1.8460</td>
<td>3.9881</td>
<td>3.9874</td>
<td>3.6179</td>
<td>3.9863</td>
</tr>
</tbody>
</table>
3. Otherwise, if no models presented in the SA matrix satisfies E, the breakpoints of a combination of nodes need to be increased. To find the best combination, first the nodes that have a “positive” effect on the MSE are identified. A positive effect means by increasing the number of breakpoints in a specified node, the MSE value is decreased from the base case MSE value, which is 3.9874 in this example. Based on Table 4.2, the nodes that have a positive effect are nodes 1 and 4 because the associated MSE values are lower than 3.9874. These identified nodes will be ranked from the most to least positive; the most positive node indicates the one that can most significantly decrease the value of MSE from the base case MSE. Then the algorithm increases the number of breakpoints of the positive nodes one node at a time, starting with the most positive to the least positive nodes, i.e. node 1 then node 4. In the example given in Table 4.2, the combinations of number of breakpoints will be tested in the sequence of (4, 2, 2, 2, 2), (4, 2, 2, 4, 2), (6, 2, 2, 4, 2) and (6, 2, 2, 6, 2). The algorithm will return the combination with a MSE that is less than E, when it is found. If all the positive nodes reach maxBP, i.e. the sequence of (6, 2, 2, 6, 2) is reached in this example, and the value of MSE is still greater than E, then no solution can be found that satisfies the given E with the number of maxBP.

If this situation arises and no solutions can be found, the user has the options of either lowering the value of E, and/or increasing the number of maxBP, and/or increase the sampling size.
4.1.3 Extracting Rules from the Generated Enhanced-PWL-ANN models

The last step of the enhanced-PWL-ANN algorithm is identical to that of the PWL-ANN algorithm. After the activation functions of the hidden neurons in the originally trained ANN model are approximated into piecewise linear equations, the sigmoid functions are replaced with the PWL functions and the enhanced-PWL-ANN model is derived. Then rules in the form of linear functions can be extracted from the enhanced-PWL-ANN model.

4.2. Discussion: Comparison of PWL-ANN Algorithm and Enhanced-PWL-ANN Algorithm

4.2.1 Comparison of Techniques Adopted in the Algorithms

The PWL-ANN and enhanced-PWL-ANN algorithms adopt different techniques:

1. The brute force search in the PWL-ANN algorithm can only locate two breakpoints in the hidden sigmoid functions and approximates the hidden sigmoid functions by 3-piece linear equations, whereas in the enhanced-PWL-ANN algorithm, the brute-force search algorithm is extended to locate N number of breakpoints and the hidden activation functions is approximated by (N+1)-piece of linear equations. By increasing the number of breakpoints, the accuracy and fidelity of the models are improved;

2. A sampling algorithm is included as an option in the enhanced-PWL-ANN algorithm in order to manage the computation time required to locate N number of breakpoints.
3. A sensitivity analysis is implemented in the enhanced-PWL-ANN algorithm to decide the number of breakpoints applicable for each of the hidden nodes in the original trained ANN model in order to help balance the trade-off between fidelity and comprehensibility of the extracted rule set.

4.2.2 Comparison of Performances

In order to demonstrate the improvement made by the enhanced-PWL-ANN algorithm on the PWL-ANN algorithm presented in Chapter 3, the enhanced-PWL-ANN algorithm was applied to the three dataset for which the preliminary PWL-ANN algorithm failed to provide satisfactory results. The characteristics of these datasets are summarized in Table 4.3. Approach 3 of the application of the PWL-ANN algorithm is used for comparison because this approach demonstrated the highest accuracy among the three approaches reported in. A comparison of parameter settings for generating the PWL-ANN and the enhanced-PWL-ANN models are summarized in Table 4.4. The discussion that follows focuses on comparing the accuracies and fidelities of the models, as well as the comprehensibility of the rule sets given by the models.
### Table 4.3 – Characteristics of the datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Data size</th>
<th>No. of input nodes</th>
<th>No. of hidden nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>#6 Yacht Hydrodynamics (UCI)</td>
<td>308</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>#7 Bodyfat (Weka)</td>
<td>252</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>#15 CPU (UCI)</td>
<td>209</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

### Table 4.4 – Parameter settings of the PWL-ANN and the enhanced-PWL-ANN models

<table>
<thead>
<tr>
<th>Parameter settings</th>
<th>PWL-ANN models</th>
<th>Enhanced-PWL-ANN models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of breakpoints (maxBP)</td>
<td>2</td>
<td>Maximum of 6</td>
</tr>
<tr>
<td>Minimum distance between breakpoints (l)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Sample data size ((D_{sampling}))</td>
<td>N/A</td>
<td>300</td>
</tr>
<tr>
<td>Desired error rate in terms of MSE percentage difference, (E)</td>
<td>&lt; 20%</td>
<td>&lt; 20%</td>
</tr>
</tbody>
</table>
4.2.2.1 Comparing Accuracy and Fidelity of the Generated Models

The values of $R^2$ and mean square error (MSE) were used to compare the accuracies of the models generated using the enhanced-PWL-ANN algorithm with those given by the PWL-ANN algorithm. Since a lower MSE value indicates less deviation of the predicted from the actual values, and a $R^2$ value close to 1 indicates that the fitted line is close to the actual values, therefore a low MSE value and a $R^2$ as close as possible to 1 are desired. Since the primary evaluation criterion of the PWL-ANN models is fidelity, the primary evaluation criterion of the enhanced-PWL-ANN models is also fidelity. A high fidelity model means that the generated model produces similar values of $R^2$ and MSE as those produced by the ANN models. To evaluate fidelity of the generated models given by both the PWL-ANN and the enhanced-PWL-ANN algorithm to the originally trained ANN model, the values of mean square error (MSE) given by the generated models and the ANN models are compared by means of the percentage difference between the values.

Table 4.5 summarizes the MSE and $R^2$ given by the three modeling efforts using: (i) the originally trained ANN models (called model #1), (ii) the PWL-ANN models (called model #2) and (iii) the enhanced-PWL-ANN models (called model #3). It can be observed from Table 4.5 that MSE values given by Model #3 are much lower than that given by Model #2 and the values of $R^2$ given by Model #3 is closer to 1 than those given by Model #2. This demonstrates an improvement in terms of accuracy given by the enhanced-PWL-ANN algorithm. Although Model #1 has the highest accuracy, with the
lowest MSE values and R^2 closest to 1, the accuracies given by Model #3 are close to that
given by Model #1 and all the R^2 values are within 0.001 differences.

**Table 4.6** summarizes the MSE percentage differences between Model #1 and Model #2,
and Model #1 and Model #3 and the number of breakpoints per node of these models.
Using Dataset #6 as an example, the number of BPs per node for Model #3 is shown as
(6, 2, 2, 4, 2), which means the model has 5 nodes, and node 1 has 4 BPs, nodes 2, 3 and
5 all have 2 BPs, and node 4 has 6 BPs.

For all the datasets, it can be seen from Table 4.6 that the percentage differences of the
datasets between Model #1 and Model #2 are over 20% and do not satisfy the desired
error rate. However, by increasing the number of breakpoints to a maximum of 6 using
the enhanced-PWL-ANN algorithm, the percentage differences for these datasets drop to
below 20%. Specifically, for dataset #6, it dropped from 164.96% to 10.45%, for dataset
#7, it dropped from 21.66% to 7.11% and for dataset #15, it dropped from 168.77% to
10.44%. Hence it can be concluded that the enhanced-PWL-ANN algorithm achieves its
objective of enhancing and improving the fidelities of the generated models such that
compared with the originally trained ANN model, their MSE percentage differences are
all below 20%.
Table 4.5 – MSE and $R^2$ given by Model #1, Model #2 and Model #3

<table>
<thead>
<tr>
<th></th>
<th>MSE</th>
<th></th>
<th></th>
<th>$R^2$</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Model #1</td>
<td>Model #2</td>
<td>Model #3</td>
<td>Model #1</td>
<td>Model #2</td>
<td>Model #3</td>
</tr>
<tr>
<td>#6</td>
<td>1.49E+00</td>
<td>3.95E+00</td>
<td>1.65E+00</td>
<td>0.9935</td>
<td>0.9828</td>
<td>0.9928</td>
</tr>
<tr>
<td>#7</td>
<td>6.26E-01</td>
<td>7.61E-01</td>
<td>6.70E-01</td>
<td>0.9910</td>
<td>0.9891</td>
<td>0.9904</td>
</tr>
<tr>
<td>#15</td>
<td>1.78E+01</td>
<td>4.78E+01</td>
<td>1.96E+01</td>
<td>0.9993</td>
<td>0.9980</td>
<td>0.9993</td>
</tr>
</tbody>
</table>

Table 4.6 – Comparison of 3 datasets: MSE percentage difference given by PWL-ANN models and Enhanced-PWL-ANN models

<table>
<thead>
<tr>
<th></th>
<th>MSE % difference between Model #1 and Model #2</th>
<th>No. of BP per node</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Model #2</td>
<td>Model #3</td>
</tr>
<tr>
<td>#6</td>
<td>164.96%</td>
<td>10.45%</td>
</tr>
<tr>
<td>#7</td>
<td>21.66%</td>
<td>7.11%</td>
</tr>
<tr>
<td>#15</td>
<td>168.77%</td>
<td>10.44%</td>
</tr>
</tbody>
</table>
It can also be observed from Table 4.6 that compared with Model #1 the accuracies and fidelities of Model #3 were improved from Model #2 due to the fact that the number of breakpoints for some of the hidden node functions was increased. As shown in Table 4.6, the MSE percentage difference of dataset #7 given by Model #2, where the number of BP was 2, was at the borderline value of 21.66%. By increasing the number of breakpoints to 4 in node 3 of the hidden nodes, Model #3 was able to decrease the MSE percentage difference compared with Model #1 from 21.66% to 7.11%. Since the MSE percentage difference compared with Model #1 of dataset #6 and #15 given by Model #2 are more than 150 times greater than their associated ANN model (Model #1), it was necessary to increase the breakpoints of more than one node of these models in order to achieve satisfactory results. For dataset #6, the number of breakpoints in nodes 1 and 4 of the originally trained ANN models were increased to 6 and 4, respectively. As a result, the percentage difference is decreased from 164.96% to 10.45%. Similarly, the number of breakpoints in nodes 1 and 4 of the originally trained ANN model of dataset #15 were increased to 4 and 6, respectively, and as a result, the percentage difference is decreased from 168.77% to 10.44%. All three cases demonstrate that by improving the accuracies of the PWL approximation on selected hidden nodes, the accuracies of the enhanced-PWL-ANN models are improved. For datasets #6 and #15, it can be seen that when fidelity is extremely low i.e. when the MSE percentage difference is over 150%, the breakpoints of a combination of nodes need to be increased in order to achieve the desired fidelity of the enhanced-PWL-ANN models (Model #3) to the original ANN models (Model #1).
4.2.2.2 Comparing Comprehensibility of the Extracted Rules from the Generated Models

In this section, the comprehensibility of the extracted rule sets given by the PWL-ANN algorithm (Model #2) and the enhanced-PWL-ANN algorithm (Model #3) are examined and compared. The objective of the analysis is to investigate the trade-off between fidelity and comprehensibility of the rule sets.

Table 4.7 compares the MSE percentage differences between (i) the originally trained ANN models (Model #1) and the PWL-ANN models (Model #2), and (ii) the originally trained ANN models (Model #1) and the enhanced-PWL-ANN models (Model #3). The number of equations extracted from the respective models are also compared.

The tradeoff between fidelity and comprehensibility is significant because while increasing the number of PWL equations per hidden node in the enhanced-PWL-ANN algorithm can improve the accuracy and fidelity of the generated rule sets, the potential number of rules will also increase significantly. An increased number of generated rules would in turn make the rule set harder to understand. For example, assuming there are \( j \) number of hidden nodes, by increasing the PWL equations from 3-piece to 5-piece, the possible number of multiple linear equations for the model increases from \( 3^j \) to \( 5^j \). The sensitivity analysis is included in the enhanced-PWL-ANN algorithm to help determine which nodes are more significant and most effective for achieving the desired error rate and enhancing fidelity of the PWL-ANN model. However, even by limiting the number of hidden nodes with increased breakpoints, it is inevitable that the number of extracted...
rules will increase. Therefore, it is necessary to examine the trade-off between fidelity and comprehensibility.

As shown in Table 4.7, Model #2 for dataset #7 was at the borderline case with a MSE difference of 21.66%. The enhanced-PWL-ANN algorithm improved the fidelity of this dataset by increasing the number of breakpoints in node 3 to 4 breakpoints. As mentioned earlier, assuming there are $j$ hidden nodes, the maximum number of extracted equations is $3 \times 3 \times 3 \ldots j$ times, i.e. $3^j$. With one of the nodes replaced by 4 breakpoints the PWL equation of that node becomes a 5-piece linear equation and the maximum number of extracted equations becomes $5 \times 3 \times 3 \times 3\ldots (j-1)$ times, i.e. $(5 \times 3^{j-1})$. Therefore, potentially, the number of extracted rules can be increased by 1.67 times:

$$\frac{5 \times 3^{j-1}}{3^j} = \frac{5}{3} = 1.67$$

After applying the enhanced-PWL-ANN model to dataset #7 thirty-five rules were generated. Since the size of the rule set given by Model #2 was 29, the increase in rules from 29 to 2.5 is only 1.2 times and this does not significantly affect comprehensibility of the rule set. However, in a case which demonstrates a borderline error rate with a larger rule sets, the user will need to decide if the trade-off between fidelity and comprehensibility is justified.
Table 4.7 – Fidelity Vs Comprehensibility for Datasets #6, #7 and #15

<table>
<thead>
<tr>
<th>Model #2</th>
<th>Model #3</th>
<th>No. of BP per nodes</th>
<th>No. of Equations Extracted</th>
</tr>
</thead>
<tbody>
<tr>
<td>#6</td>
<td>164.96%</td>
<td>10.45%</td>
<td>6, 2, 2, 4, 2</td>
</tr>
<tr>
<td>#7</td>
<td>21.66%</td>
<td>7.11%</td>
<td>2, 2, 4, 2, 2, 2, 2, 2, 2</td>
</tr>
<tr>
<td>#15</td>
<td>168.77%</td>
<td>10.44%</td>
<td>4, 2, 2, 6, 2</td>
</tr>
</tbody>
</table>
Both Model #2 of dataset #6 and #15 have a MSE percentage differences above 150%, and the breakpoints of more than one node were increased for both datasets in order to improve the performance of the enhanced-PWL-ANN models. As shown in Table 4.7, both datasets had one node increased to 4 breakpoints, and one node increased to 6 breakpoints. With this higher number of breakpoints, the potential maximum number of equations for both datasets becomes \((7 \times 5 \times 3^{j-2})\), and the potential rule set size could have been increased by 3.98 times:

\[\frac{7 \times 5 \times 3^{j-2}}{3^j} = \frac{7 \times 5}{9} = 3.98\]

As observed from Table 4.7, the number of rules was doubled for both datasets when more than one breakpoints were increased. The number of rules for dataset #6 was increased from 16 in Model #2 to 32 in Model #3, and the number of rules for dataset #15 is increased from 9 in Model #2 to 18 in Model #3. These two datasets provide good examples for which considerations of the trade-off between fidelity and comprehensibility is important. The user may decide that the original rule set is relatively small, then even though the number of rule is doubled, the size of the newly generated rule set is still acceptable, and a good balance between fidelity and comprehensibility is achieved. However, if the user considers the original rule set is large, then the user can decide whether this increase in the size of the rule set is justified. For each of the datasets #6 and #15, it is likely the user would decide the cost of doubling the number of rules is justified because the fidelity of the models is significantly enhanced, from over 160% to just over 10%.
4.3. “Opening the Black Box”: Extracting Rules from the Generated Models

In this section, Dataset #6 is used to illustrate how the extract equations can be translated into rules in order to enhance the understanding of the relationships between the predictor and predicted attributes. Three sets of rules were generated: (1) the original rule set given by the enhanced-PWL-ANN model, (2) a set of rules given by the classification tree algorithm using the equations generated by the enhanced-PWL-ANN model, and (3) a set of rules given by the regression tree algorithm using the original dataset. The first two rule sets demonstrate how to translate the equations given the enhanced-PWL-ANN model into IF-THEN rules in order to gain understanding to the problem domain. The decision tree algorithm was employed in order to simplify and improve the comprehensibility of the generated rule set. Since the decision tree algorithm can also solve regression problems, the last rule set was generated for performance comparison.

Obtained from the UCI repository (Lichman, 2013), Dataset #6 consists of 6 continuous input or predictor attributes:

1. Longitudinal position of the center of buoyancy (labeled as x1),
2. Prismatic coefficient (labeled as x2),
3. Length-displacement ratio (labeled as x3),
4. Beam-draught ratio (labeled as x4),
5. Length-beam ratio (labeled as x5), and
6. Froude number (labeled as x6).
The predicted variable is the residuary resistance per unit weight of displacement (labeled as $y_1$).

As shown in Table 4.7, a total of 32 rules or equations were generated for this dataset. Each rule given by the enhanced-PWL-ANN model is accompanied by the range of values the rule covers for each attribute, and the ranges of values provide the IF- or pre-conditions in each rule. Each rule can be expressed as follows:

Rule #1:

$IF$

(-1 $\leq$ Longitudinal position of the center of buoyancy ($x_1$) $\leq$ 1) AND
(-1 $\leq$ Prismatic coefficient ($x_2$) $\leq$ 1) AND
(-1 $\leq$ Length-displacement ratio ($x_3$) $\leq$ 1) AND
(-0.25984 $\leq$ Beam-draught ratio ($x_4$) $\leq$ 1) AND
(-1 $\leq$ Length-beam ratio ($x_5$) $\leq$ 0.2967) AND
(-1 $\leq$ Froude number ($x_6$) $\leq$ -0.53846)

$THEN$

$y_1 = -0.00121x_1 - 0.00725x_2 + 0.004772x_3 - 0.00422x_4 - 0.00603x_5 + 0.09342x_6 - 0.88651$

The complete set of rules given by the enhanced-PWL-ANN model and range of values each rule covers is shown in Appendix A Figure A1 and Figure A2, respectively.
The analysis in (Chan & Chan, 2017a) showed that knowledge about the problem domain can be obtained by examining the coverage of each rule in order to identify the more significant rules, and by studying the magnitude of coefficients of the attributes and the more significant attributes can be discovered. However, Chan & Chan (2017a) pointed out two major areas that needed to be improved. First, the comprehensibility of extracted rule set can be improved by removing the rules that have extremely low coverage. Second, the extracted rules are expressed in terms of normalized values and in order to apply the rules, the user has to normalize the data values and then denormalize the values given the rules.

In this illustrative example, an attempt was made to improve the first area. The decision tree algorithm was applied in order to identify the IF-THEN conditions and to simplify the rule set. When applying the decision tree algorithm, the problem becomes a classification problem in which the predictor or input attributes are the same as those used to generate the enhanced-PWL-ANN model but the predicted or output attribute is the rules given by the enhanced-PWL-ANN model. The classification tree is obtained using the \texttt{rpart()} function implemented in R and it is shown in Figure 4.3.
Figure 4.3 – Classification tree obtained for dataset #6 using the rules from the generated enhanced-PWL-ANN model as predicted attributes
It can be seen from Figure 4.3 that not all the rules or equations given by the enhanced-PWL-ANN model were used in the classification tree; the equations that were used can be identified from the leaves of the decision tree and they are Rule #1, #2, #3, #5, #6, #7, #8, #9, #12, #14, #15, #18, #20, #22. Starting from the far left branch, the rule for the first leaf can be read as

\[ IF (x6 \geq 0.77) \text{ AND } (x7 \geq 0.22) THEN y1 = \text{ Equation #7 in Figure B1}. \]

The rule given by the second leaf can be read as:

\[ IF (x6 \geq 0.77) \text{ AND } (x7 < 0.22) THEN y1 = \text{ Equation #9 in Figure B1}. \]

By counting the number of leaves given by the classification tree, the total of 16 rules was generated. The complete set of rule is shown in Appendix B.

Since the decision tree algorithm can also solve regression problems, a regression tree was also generated for comparison. The predictor and predicted variables used to generate the enhanced-PWL-ANN model were used to train the regression tree model. **Figure 4.4** shows the trained regression tree model using \textit{rpart()} in R and a total of 5 rules were given, with x6 as the only predictor attribute used in the pre-conditions.

The rules given by the regression tree are as follows:

1. IF \((x6 < 0.31)\) THEN \(y1 = -0.94\)
2. IF \((0.31 \leq x6 < 0.62)\) THEN \(y1 = -0.66\)
3. IF \((0.62 \leq x6 < 0.77)\) THEN \(y1 = -0.29\)
4. IF \((0.77 \leq x6 < 0.92)\) THEN \(y1 = 0.12\)
5. IF \((x6 \geq 0.92)\) THEN \(y1 = 0.67\)
Figure 4.4 – Regression tree obtained for dataset #6
The set of rules given by the enhanced-PWL-ANN model, classification tree and regression tree are called Rule Set A, Rule Set B and Rule Set C, respectively. The predictive accuracies in terms of MSE and $R^2$ and the comprehensibility in terms of number of rules and number of pre-conditions of rule sets are summarized in Table 4.8. Note that the values in the pre-conditions and the predicted values given by all three rule sets are still in terms of normalized values.

Comparing Rule Set A and Rule Set B, Rule Set B is more simplified in terms of both the pre-condition of the rules and the size of the rule set. A total of 16 rules were given by Rule Set B with 2 to 4 pre-conditions per rule, whereas Rule Set A has 32 rules with 6 pre-conditions each. Therefore, Rule Set B has better comprehensibility than Rule Set A. However, as a trade-off, Rule Set B is slightly less accurate than Rule Set A. Rule Set A has a MSE of 1.65 and a $R^2$ of 0.9928, whereas Rule Set B has a MSE and $R^2$ of 1.76 and 0.9923 respectively. In comparison, Rule Set C gave the least amount of rule and pre-conditions; a total of 5 rules with 1 pre-condition each were given by Rule Set C. However, it also has the lowest accuracy with the highest value of MSE and lowest value of $R^2$. Comparing to Rule Set A and B given by the enhanced-PWL-ANN model, the MSE of the rule set given by the regression tree approach was 3.2 and 3.0 times higher, respectively.
Table 4.8 – Comparing accuracies and comprehensibility of Rule Set A, Rule Set B and Rule Set C

<table>
<thead>
<tr>
<th></th>
<th>Rule Set A</th>
<th>Rule Set B</th>
<th>Rule Set C</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Accuracy</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSE</td>
<td>1.65</td>
<td>1.76</td>
<td>5.24</td>
</tr>
<tr>
<td>R²</td>
<td>0.9928</td>
<td>0.9923</td>
<td>0.9771</td>
</tr>
<tr>
<td><strong>Comprehensibility</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No. of rules</td>
<td>32</td>
<td>16</td>
<td>5</td>
</tr>
<tr>
<td>No. of pre-conditions per rule</td>
<td>6</td>
<td>2 – 4</td>
<td>1</td>
</tr>
</tbody>
</table>
4.4. Conclusion

This chapter presents an enhanced version of the PWL-ANN algorithm proposed in Chapter 3. The enhanced-PWL-ANN algorithm is able to generate models with higher accuracy and higher fidelity to the originally trained ANN models. The modifications made to the preliminary PWL-ANN algorithm include the following:

1. In order to improve accuracy and fidelity of the PWL-ANN model, the enhanced brute force search is applied to locate N number of breakpoints in the enhanced-PWL-ANN algorithm and the hidden activation functions is approximated by (N+1)-piece of linear equations;

2. In order to manage computation time of the algorithm, a sampling algorithm is included as an option in the enhanced-PWL-ANN algorithm;

3. In order to help balance the trade-off between fidelity and comprehensibility of the extracted rule set, a sensitivity analysis is implemented to decide the number of breakpoints applicable for each of the hidden nodes in the original trained ANN model in the enhanced-PWL-ANN algorithm.

The enhanced-PWL-ANN algorithm is tested on the 3 datasets that had unsatisfactory results in Chapter 3 and the accuracy, fidelity and comprehensibility of the models are compared. Based on the analysis, the following conclusions can be drawn:

1. By increasing the number of the breakpoints in the selected hidden nodes of the original ANN model, the fidelity of the enhanced-PWL-ANN models derived from the datasets with previously unsatisfactory results from the PWL-ANN models are all improved. The new models from application of the enhanced-
PWL-ANN algorithm have MSE differences that are less than 20% compared to the originally trained ANN models.

2. There is a clear trade-off between the fidelity of the enhanced-PWL-ANN models and the comprehensibility of the generated rule sets. Depending on the characteristics of the dataset, the user needs to consider and determine if increasing the breakpoints of select hidden nodes is justified.

A detail illustrative example using Dataset #6 was given to demonstrate how to translate the equations/rules extracted by the enhanced-PWL-ANN algorithm into understandable IF-THEN rules. In the example, three sets of rules were generated using (1) the equations given by the enhanced-PWL-ANN model, (2) the equations given by the enhanced-PWL-ANN model combined with classification decision tree algorithm, and (3) regression decision tree algorithm. The result showed that both sets of rules generated based on the enhanced-PWL-ANN algorithm outperformed the set of rules given by the regression tree algorithm. By applying the decision tree algorithm onto the rules generated by the enhanced-PWL-ANN algorithm, the less significant rules were removed and the comprehensibility of the rule set was improved. However, this improvement in comprehensibility can compromise the accuracy of the generated rule set.

In addition to enhancing understanding of the problem domain, the extracted rules can be used for specific problem solving cases. For example, the user may need to achieve a target output, \( Y \), and all the predictor parameters are known except for two parameters, \( X_1 \) and \( X_2 \). The problem then becomes one of determining the values of the unknown predictor attributes that would satisfy the target output value. To determine the two
unknown parameter values, the equations that cover the values of the known attributes can be identified and used for solving for the unknown parameters. The algorithm to support this scenario of back calculation will be considered as the next step of this study.
CHAPTER 5 THE EURECA ALGORITHM

In this chapter, an eclectic artificial neural network (ANN) rule extraction algorithm called EURECA (Extracting Understandable Rules by Employing Clustering with Artificial Neural Network) is proposed. The proposed EURECA algorithm can predict as least as accurately as the trained ANN modes. To overcome the opaque nature of the trained ANN models, the EURECA algorithm can (i) generate interpretable equations from the model so as to provide some insights about the relationships between the predictor and predicted attributes, and (ii) shed light on the relationship among the data tuples that are assigned to a similar range of weighted input values.

5.1 Motivation and Heuristic behind the Proposed Algorithm

The proposed EURECA algorithm groups the data tuples according to their weighted input values. This heuristic is inspired by the PWL-ANN algorithm described in Chapter 3, which extracts rules in the form of multiple linear regression equations by approximating the hidden activation function into some piecewise linear equations. The PWL-ANN algorithm generated rule sets with accuracies close to those given by the originally trained ANN models. Based on the fact that a given problem domain can be modeled by a set of linear equations, we posed the hypothesis that some linear relationships are present among attributes within the dataset. Since the limits of the line segments in the piecewise linear equations are defined by some weighted input values, it is assumed that the linear relationships within the dataset are also defined by the weighted input values. In other words, it was hypothesized that during the learning process, the ANN algorithm managed to identify data tuples which shared some properties through the weight adjustment process. As the ANN model adjusts the weights between the input
and hidden layer neurons, it assigns the same range of weights to the data tuples with shared properties and one of the properties shared by the data tuples within the same subset is a linear relationship.

Another heuristic behind the proposed algorithm was based on general observation that refining a given dataset to smaller subgroups usually improves the accuracy of a given machine learning algorithm applied to the dataset. For example, Huang et al. (2015) improved the predictive accuracy of the ANN model on the level of phytoplankton biomass in Lake Poyang, China, by dividing the dataset into upstream and downstream subsets using k-mean clustering. Zhu et al. (2016) improved the accuracy of their ANN model for predicting credit risk of enterprises in supply chains by identifying the significant variables using logistic regression and then training the ANN models with the identified variables. Ebtehaj et al. (2016) partitioned the dataset using the decision tree approach and trained each subset with the ANN algorithm. As a result, the combined ANN models from the subsets of data outperformed the ANN model that was trained using the entire dataset. Hence, in the design of EURECA, it was also assumed that subdividing the original dataset according to the weights of the hidden neurons will produce generated models that can predict more accurately than the ANN algorithm which was applied to the entire dataset. Therefore, it was hypothesized that the EURECA algorithm will have higher predictive accuracy compared to the ANN and PWL-ANN algorithm.
On top of better predictive accuracy, another advantage EURECA has over the PWL-ANN algorithm is that EURECA always generates equations with physical meaning in the problem domain. This is in contrast to the PWL-ANN algorithm, which produces output equations that depend on the dataset used to train the original ANN models. Therefore, if the dataset that trained the original ANN model consists of normalized values, which is often the case, then the equations generated by the PWL-ANN algorithm will be expressed in terms of normalized values. As a consequence, these equations do not have physical meaning and are not interpretable by the domain experts. When this happens, the domain experts who use the equations for problem-solving need to perform the additional steps of pre-processing of the dataset, and post-processing of the dataset and generated equations. While these extra steps are not complicated and are in fact necessary to give meaning to the generated equations, they can be inconvenient for the domain experts.

5.2 The EURECA Algorithm

The proposed methodology of EURECA, which is an eclectic ANN rule extraction algorithm, consists of three main steps:

1. Train the ANN model,
2. Cluster the dataset according to the hidden neuron weight values, and
3. Perform MLR on each of the clustered group of data.

The three steps in the proposed EURECA methodology will be discussed as follows.
5.2.1 Training the ANN model

In the first step of the EURECA algorithm, any feedforward neural network algorithm can be used for training the ANN model. There are two requirements in developing the trained ANN model: (i) the ANN model should involve a feedforward ANN containing only one hidden layer and the output layer activation function should be a linear function, (ii) the weights of the hidden nodes of the trained ANN model are used in step 2 of the EURECA methodology for clustering the data points in the given dataset.

5.2.2 Clustering the dataset according to the hidden neuron weight values

In the second step of the methodology, the dataset is divided into groups according to the associated weighted input values of each hidden node. The weighted input of the dataset to a particular hidden node is derived by multiplying the predictor or input values by the weight values of the hidden neuron. Consider the hidden neuron, $j$, as shown in Figure 5.1.
Figure 5.1 – Hidden neuron, $j$
Assuming there are \( n \) predictor variables, the weighted input, \( \xi_j \), to neuron \( j \) is given by:

\[
\xi_j = \sum_{i=1}^{n} w_{ij}x_i + \theta_j
\]

where \( x_i \) is the value of input \( i \),

\( w_{ji} \) is the weight of input \( i \) to node \( j \), and

\( \theta_j \) is the bias value of node \( j \).

Note that the weighted input is calculated using the normalized input values since the ANN models are trained with normalized input values.

The dataset is divided by some user-defined weighted input values, called breakpoints. For example, if the user-defined breakpoints are \((-\xi, \xi)\), then the dataset that feeds into the hidden neuron \( j \) can be divided into three groups: group A is when \( \sum_{i=1}^{n} w_{ij}x_i + \theta_j < -\xi \); group B is given when \(-\xi \leq \sum_{i=1}^{n} w_{ij}x_i + \theta_j \leq \xi \); and group C is given when \( \sum_{i=1}^{n} w_{ij}x_i + \theta_j > \xi \), as shown in Figure 5.2. But this happens within each hidden node in the ANN model. Therefore, from the macro perspective shown in Figure 5.3, each data tuple, which consists of 1 set of input parameter values, is assigned to a group according to the weighted input value \( (w_{ij}x_i) \) in each of the hidden nodes, and the final clusters of the dataset are given by the combinations of the hidden neuron groups.

For example, consider a trained ANN with three hidden neurons as shown in Figure 5.3. The data tuple has a weighted input value that belongs to group A in hidden neuron 1, group B in hidden neuron 2 and group A in hidden neuron 3 again, as shown by the
shaded portion of the graph in the box each hidden node in Figure 5.3. This hidden-neuron-group combination of (group A, group B, group A) is considered as Cluster 1, and all the data tuples that have the same combination will be assigned to Cluster 1. Another data tuple has a weighted input value that belongs to group A in all hidden neurons. Then, the combination of (group A, group A, group A) becomes Cluster 2, and all the data tuples that share this combination will belong Cluster 2. Each of the different combination forms its own cluster, and all the data tuples that have the same combination are assigned to that cluster.

Since the clustering of data depends only on the weighted input, the EURECA algorithm is not limited to ANNs with hidden sigmoid activation functions. In fact, the EURECA algorithm does not require as input any particular type of activation function in the hidden layer of the originally trained ANN model. That is, as long as the originally trained ANN model is a feedforward neural network, where the weighted input is given by $\sum_{i=1}^{n} w_{ij} x_i + \theta_j$, the EURECA algorithm can be employed.
Figure 5.2 – Clustering by weighted input values – dataset is clustered into three groups by the weighted input values in each hidden neuron
Figure 5.3 – Data clusters are defined by the combination of groups a dataset belongs to in each hidden neuron.
5.2.3 Perform MLR on each of the clustered group of data

When the clustering of the input data tuples according to their weighted input values is completed, the clustered dataset is modeled using MLR. The MLR equations can be generated using either the normalized or real data values. To model with the actual or physical values of the dataset, the cluster groups assigned to the normalized data tuples are mapped back to the corresponding data tuples with actual values. The output of EURECA is the set of MLR equations obtained for each of the clustered groups of real data with physical values.

5.3. Accuracy of the EURECA models

To evaluation EURECA’s performance, it was tested on ten datasets obtained from the UCI repository (Lichman, 2013). The performances of the PWL-ANN (Chan & Chan, 2017) and EURECA algorithms were compared because they both target regression problems and generate MLR equations. The accuracies of the models generated by the two algorithms were compared to that given by the originally trained ANN models.

For the ten datasets, if the dataset contains discrete attributes, the different classes of the discrete attribute were replaced by dummy variables. For example, the input attributes that can be categorized into the classes of high, medium, and low, were replaced with the dummy variables 1, 2 and 3. Table 5.1 summarizes the characteristics of the datasets, the number of input nodes and the number of hidden nodes in the trained ANN models, as well as the locations of breakpoints for the EURECA models.
<table>
<thead>
<tr>
<th>#</th>
<th>Dataset</th>
<th>Dataset size</th>
<th>No. of input nodes</th>
<th>No. of hidden nodes</th>
<th>Locations of BPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>Energy Efficiency Heat Loading</td>
<td>767</td>
<td>8</td>
<td>6</td>
<td>(-2.49, 2.49)</td>
</tr>
<tr>
<td>#2</td>
<td>Energy Efficiency Cool Loading</td>
<td>767</td>
<td>8</td>
<td>6</td>
<td>(-2.49, 2.49)</td>
</tr>
<tr>
<td>#3</td>
<td>Airfoil Self-Noise</td>
<td>1503</td>
<td>5</td>
<td>4</td>
<td>(-2.49, 2.49)</td>
</tr>
<tr>
<td>#4</td>
<td>CCPP</td>
<td>9568</td>
<td>4</td>
<td>4</td>
<td>(-2.49, 2.49)</td>
</tr>
<tr>
<td>#5</td>
<td>Housing</td>
<td>506</td>
<td>13</td>
<td>8</td>
<td>(-2.49, 2.49)</td>
</tr>
<tr>
<td>#6</td>
<td>Yacht Hydrodynamics</td>
<td>308</td>
<td>6</td>
<td>5</td>
<td>(-2.18, 2.15)</td>
</tr>
<tr>
<td>#7</td>
<td>Parkinson</td>
<td>5873</td>
<td>19</td>
<td>12</td>
<td>(-2.49, 2.49)</td>
</tr>
<tr>
<td>#8</td>
<td>Concrete</td>
<td>1030</td>
<td>8</td>
<td>6</td>
<td>(-2.49, 2.49)</td>
</tr>
<tr>
<td>#9</td>
<td>Pollution</td>
<td>60</td>
<td>15</td>
<td>9</td>
<td>(-2.49, 2.49)</td>
</tr>
<tr>
<td>#10</td>
<td>CPU</td>
<td>209</td>
<td>7</td>
<td>7</td>
<td>(-1.00, 1.00)</td>
</tr>
</tbody>
</table>
The values of mean square error (MSE) and $R^2$ were used to measure the accuracies of the generated models. The MSE measures the deviation of the predicted value from the actual value; and $R^2$ shows how close the actual values are to the fitted line. Hence, a low MSE value and an $R^2$ value that is close to 1 indicate the high accuracy of a model. The values of $R^2$ and MSE can be calculated as follows:

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

$$MSE = \frac{\sum_i (y_i - \hat{y}_i)^2}{n}$$

where $y_i$ = the expected output of the $i^{th}$ dataset, $\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$ and $\hat{y}$ = predicted output, $n$ = number of data tuples.

The values of MSE and $R^2$ given by the originally trained ANN models, the PWL-ANN models and EURECA are shown in Table 5.2.
Table 5.2 – MSE and $R^2$ given by the ANN, PWL-ANN and EURECA models

<table>
<thead>
<tr>
<th></th>
<th>MSE</th>
<th>R²</th>
<th></th>
<th>MSE</th>
<th>R²</th>
<th></th>
<th>MSE</th>
<th>R²</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>9.52E-01</td>
<td>1.10E+00</td>
<td>#2</td>
<td>3.54E+00</td>
<td>3.57E+00</td>
<td>#3</td>
<td>1.61E+01</td>
<td>1.65E+01</td>
</tr>
<tr>
<td>#4</td>
<td>2.47E+01</td>
<td>2.54E+01</td>
<td></td>
<td>2.09E+00</td>
<td>2.09E+00</td>
<td>#5</td>
<td>6.81E+00</td>
<td>7.67E+00</td>
</tr>
<tr>
<td>#6</td>
<td>1.49E+00</td>
<td>3.95E+00</td>
<td></td>
<td>1.47E+00</td>
<td>1.47E+00</td>
<td>#7</td>
<td>1.93E+01</td>
<td>1.97E+01</td>
</tr>
<tr>
<td>#8</td>
<td>5.51E+01</td>
<td>5.85E+01</td>
<td></td>
<td>2.92E+01</td>
<td>2.92E+01</td>
<td>#9</td>
<td>4.98E+02</td>
<td>5.94E+02</td>
</tr>
<tr>
<td>#10</td>
<td>1.78E+01</td>
<td>4.78E+01</td>
<td></td>
<td>1.28E+01</td>
<td>1.28E+01</td>
<td>#10</td>
<td>0.9907</td>
<td>0.9935</td>
</tr>
<tr>
<td>#2</td>
<td>1.10E+00</td>
<td>2.09E+00</td>
<td></td>
<td>0.9907</td>
<td>0.9892</td>
<td>#3</td>
<td>2.09E+00</td>
<td>2.09E+00</td>
</tr>
<tr>
<td>#5</td>
<td>6.81E+00</td>
<td>7.67E+00</td>
<td></td>
<td>6.618</td>
<td>0.6522</td>
<td>#7</td>
<td>3.84E+00</td>
<td>0.8311</td>
</tr>
<tr>
<td>#8</td>
<td>5.85E+01</td>
<td>2.92E+01</td>
<td></td>
<td>0.9150</td>
<td>0.9129</td>
<td>#9</td>
<td>1.47E+01</td>
<td>0.8311</td>
</tr>
<tr>
<td>#10</td>
<td>4.78E+01</td>
<td>1.28E+01</td>
<td></td>
<td>0.9993</td>
<td>0.9980</td>
<td>#10</td>
<td>2.92E+01</td>
<td>0.8691</td>
</tr>
<tr>
<td>#10</td>
<td>1.28E+01</td>
<td>1.28E+01</td>
<td></td>
<td>0.9993</td>
<td>0.9995</td>
<td>#10</td>
<td>2.92E+01</td>
<td>1.0000</td>
</tr>
</tbody>
</table>
Table 5.2 shows that 6 out of the 10 ANN models have $R^2$ values greater than 0.9, and the other four ANN models have $R^2$ values that range from 0.6619 to 0.8691. While the accuracies of the latter four ANN models maybe not be acceptable as predictive models, they are accepted for this study because the performances of the trained ANN models are used only for comparison to the PWL-ANN and EURECA models. Since the primary evaluation criterion in the development of the PWL-ANN algorithm was fidelity, the PWL-ANN models were designed to perform similarly as the originally trained ANN models. It can be observed from Table 5.2 that for the datasets for which the ANN models demonstrated low accuracy with $R^2$ values less than 0.9, the PWL-ANN models also had low accuracy and all the $R^2$ values of the PWL-ANN models are within a range of 0 to 0.02 from the $R^2$ values of the original ANN models. While the PWL-ANN algorithm did not outperform the ANN algorithm on any of the datasets, the EURECA algorithm outperforms both the ANN and PWL-ANN approaches on all the datasets. It can be seen from Table 5 that all of the EURECA models generated have $R^2$ values that exceed 0.88 and were greater than those given by the originally trained ANN and the PWL-ANN models. Hence it can be concluded that the EURECA algorithm generates models with higher predictive accuracy than both the ANN algorithm and the PWL-ANN algorithm. Therefore, the research objective that EURECA can achieve a higher prediction accuracy than the originally trained ANN and the PWL-ANN model is achieved.
5.4. Conclusion

An eclectic ANN rule extraction algorithm called EURECA is proposed to address the gap in research caused by the scarce work done on developing eclectic ANN rule extraction algorithm that targets regression problems. The algorithm extracts multiple linear regression equations by first segmenting the dataset into clusters according to the associated weighted input values of the hidden nodes in the originally trained ANN model. The weighted inputs are calculated by multiplying the values of the predicted attributes with their associated hidden weights. Then, MLR analysis is performed on each of the clustered groups of data. The set of MLR equations produced for each cluster is the set of rules that describes the studied problem domain. By clustering the dataset based on the associated weighted input values of the originally trained ANN model, the original dataset is divided into subsets, and each subset is hypothesized to be a set of linearly related attributes.

The rules extracted by the EURECA algorithm are expressed in terms of the real values of the dataset, which overcomes the major drawback of the enhanced-PWL-ANN algorithm caused by normalization of the dataset. The accuracy of the EURECA algorithm was verified by 10 test cases from the UCI repository. The performances given by the EURECA models were compared to those given by the originally trained ANN models and the PWL-ANN models in Chapter 3. The results showed that the models generated by the EURECA algorithm achieved higher predictive accuracy than both the ANN models and the PWL-ANN models. The EURECA algorithm was also proposed to
“open the black-box” of the ANN algorithm by investigating the relationship between the assigned hidden weight value and the data characteristic and this investigation will be presented in Chapter 7.
Decision support systems (DSS) are computer-based systems that support human decision-making in complex problems (Sharma & Yetton, 1986). The term “decision support systems” was loosely coined by Keen and Scott Morton in 1978 to describe systems that provide decision-making support to humans using information processing (Er, 1988). This chapter presents a decision support system that is developed based on the eclectic ANN rule extraction algorithm, EURECA, presented in Chapter 5. The EURECA algorithm translates the complex non-linear relationships implicit in the ANN model into a set of multiple linear regression equations which are easier to understand and apply. Most importantly, the equations provide some insight into the relationships between the predictor and predicted parameters of the trained ANN model. In this way, explicit knowledge about the studied problem can be obtained.

The overall objective of this chapter is to present the development and application of EURECA decision support system. The main objectives of the DSS are:

1. To generate rules in terms of multiple linear equations based on a given set of data on a studied problem, the set of rules is called the EURECA model,
2. To provide some insight to the decision maker on the relationships between the predictor and predicted attributes in the given dataset,
3. To support users to perform back calculation in a problem-solving scenario related to the studied problem, which requires determining the values of the unknown predictor attributes so that the target predicted values could be achieved, and
4. Based on the EURECA model, to assist decision makers to generate accurate predictions given a new set of predictor parameters.

The DSS provides the following functions to the user to achieve the four objectives:

1. For objective #1, the DSS assists the user in:
   a. Uploading the dataset and the trained ANN model in csv format to generate the EURECA model
   b. Specifying the desired number of breakpoints for the EURECA model.

2. For objective #2, the DSS assists the user in:
   a. Evaluating the accuracy and significance of the generated model and rule set,
   b. Evaluating the significance of each generated rule, and
   c. Evaluating the relative significance of the predictor attributes on the predicted attributes.

3. For objective #3, the DSS assists the user in solving the back calculation problem using either the novice or expert approach.

4. For objective #4, the DSS assists the user in uploading new input dataset and using the generated EURECA model for prediction.

6.1. EURECA DSS

The schematic overview of the EURECA algorithm and the EURECA DSS is shown in Figure 6.1. Apart from generating rule sets using the EURECA algorithm, the DSS also performs analysis on the extracted rules and the predictor attributes, supports solving the
back calculation problem, and predicts output values when given a new set of input attributes.

6.1.1 Structural Framework of the DSS

The structural framework of the DSS is shown in Figure 6.2. The EURECA DSS consists of six modules as shown in upper half of Figure 6.2, which include (1) graphical user interface (GUI) module, (2) data input/output (I/O), (3) the EURECA modeling module, (4) the rules and data evaluation module, (5) the back calculation module, and (6) the prediction module. The algorithms and approaches that support each of these modules (bottom half of Figure 6.2) are described in the next section.
Figure 6.1 – Schematic diagram of the EURECA algorithm and the EURECA decision support system
Figure 6.2 – Structural framework of the EURECA DSS
6.1.2 Detailed Methodologies of the DSS

6.1.2.1 The EURECA algorithm

The EURECA algorithm is responsible for generating a rule set based on a given ANN model. It is an eclectic ANN rule extraction algorithm that forms the basis of the EURECA DSS. The main objective of the EURECA algorithm is to extract rules in the form of multiple linear regression equations from trained ANN models. As shown in Figure 6.1, the EURECA algorithm consists of three main steps: (1) training the ANN model (2) clustering the dataset, and (3) performing multiple linear regression (MLR) on each of the clustered groups of data tuples. The details of the EURECA algorithm and its performance can be found in Chapter 5.

6.1.2.2 Model and Parameters Evaluation

The EURECA DSS supports the decision maker in the analysis of the extracted rules and evaluation of the performance the generated EURECA model in terms of accuracy and significance. It also analyzes the significance of the predictor attributes in relation to the predicted attributes by analyzing the attributes’ coefficients. The indicators and approaches used in the analysis process are described as follows.

A. Accuracy and Significant of EURECA Model

The accuracy of the generated EURECA model is measured by the value of the mean squared error (MSE) and $R^2$. MSE measures the deviation of the predicted value from the actual value, and $R^2$ shows how close the actual values are to the fitted line. The lower
the MSE value, the lower the deviation of the predicted from the actual values, and the closer the R² value to 1, the closer the fitted line is to the actual values. The values of R² and MSE can be calculated as follows:

\[
R^2 = 1 - \frac{\sum_i(y_i - \hat{y}_i)^2}{\sum_i(y_i - \bar{y})^2}
\]

\[
MSE = \frac{\sum_i(y_i - \hat{y}_i)^2}{n - p - 1}
\]

where \( y_i \) = the expected output of the \( i \)th data tuple, \( \hat{y} \) = average output value = \( \frac{1}{n} \sum_{i=1}^{n} y_i \), \( \hat{y} \) = predicted output, \( n \) = number of data tuples and \( p \) = number of predictor attributes.

The analysis of variance (ANOVA) table is used to demonstrate the significance of the generated rule set. The analysis of variance is a common statistical practice used to evaluate the significance of a linear regression model and is included in commercially available software such as Matlab® (The MathWorks, Inc., 2016) and SPSS (IBM, 2016). The ANOVA table provides indicators on the performance of a model, and they include the indicators of sum of squares regression (SSR), sum of squares error (SSE), sum of squares total (SST), degree of freedom, mean squared regression (MSR), mean squared error (MSE), F ratio and p value. These indicators are explained as follows and a summary of equations from all the indicators, except the P value, is given in Table 6.1.
Table 6.1 – Equations of Indicators of the ANOVA table

<table>
<thead>
<tr>
<th></th>
<th>Sum of Squares</th>
<th>Degree of Freedom</th>
<th>Mean Squares</th>
<th>F Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Regression</strong></td>
<td>( SSR = \sum_i (\hat{y}_i - \bar{y})^2 )</td>
<td>( p )</td>
<td>MSR = SSR/p</td>
<td>( F = \frac{MSR}{MSE} )</td>
</tr>
<tr>
<td><strong>Error</strong></td>
<td>( SSE = \sum_i (y_i - \hat{y}_i)^2 )</td>
<td>( n - p - 1 )</td>
<td>MSE = SSE/(n-p-1)</td>
<td>---</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>( SST = SSR + SSE )</td>
<td>( n - 1 )</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>

where \( y_i \) = the target output of the \( i^{th} \) data tuple, \( \bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i \) = average value of target output, \( \hat{y}_i \) = predicted output of the \( i^{th} \) dataset, \( n \) = size of dataset, and \( p \) = number of attributes,
The SSR is the sum of squares of the deviations between the predicted values and the mean value of the actual values. The SSE is the sum of squares of the deviation between the predicted value and the actual value. The SST is the sum of total and is given by the sum of SSE and SSR. The MSR is given by dividing the SSR with the number of attributes and MSE is given by dividing the SSE with the difference between the number of data tuple and number of predictor attributes minus 1. All of these values measure the deviation between the generated model and the studied dataset, and the goal is to minimize these values as much as possible. The F ratio examines the significance of the relationship between the predictor variables given by the rule set and the target predictor variables. The larger the value of F, the stronger the relationship; a value of greater than 1 is desirable. The P value reflects the likelihood of the predictor value given by the generated model being as significant as F. Usually, a value of p < 0.05 is a good indication that the generated model is likely to be as significant as the value indicated by F. The P value given in the ANOVA table in the EURECA DSS is calculated using the \( pf() \) function provided in R (R Core Team, 2015).

\[ \text{Identifying Most Significant Predictor Attributes} \]

The EURECA DSS supports the user in understanding the studied dataset by revealing the significance of the input attributes. The EURECA DSS analyzes the significance of each attribute in an extracted rule by calculating the product of the attribute’s coefficient and the maximum value the attribute covers for a given rule. The attribute with the largest mathematical product is the most dominating attribute for a given rule. The DSS then
ranks the dominating attributes according to their coverage in the problem space. The purpose of this analysis is to provide statistical insight on the relative significances among the input attributes of the dataset of the problem domain.

6.1.2.3 Back Calculation Algorithms

The back calculation problem arises when the user has a target predicted values that need to be achieved, but only partial information on the values of the predictor attributes is available. The task is then to determine the unknown predictor attributes values such that the target predicted values could be achieved. The EURECA DSS supports problem-solving in the back calculation scenario through two approaches: (1) the novice approach and (2) the expert approach. In the novice approach, the DSS searches for the values of the unknown predictor attributes using a brute-force search and returns to the user the best solution based on the calculated error. In contrast, in the expert approach, the user is assumed to be more knowledgeable and able to select the equations to be used and provide input on the likely values of the unknown predictor attributes. The detail of the back calculation approaches is outside the scope of this paper.

6.1.2.4 Prediction Based on Unseen Dataset

The EURECA DSS supports the user to perform prediction based on a new input from the same domain. When a new dataset is presented to the system, the system generates clusters based on the new dataset according to the weighted input using the previously learned hidden neuron weights and identifies the cluster each data tuple belongs. The data tuples in the new dataset are then divided into clusters, and the DSS calculates the
predicted output values using the rules given by the generated EURECA model for the identified clusters.

6.2. Implementation of the EURECA DSS

The EURECA DSS assumes as input trained ANN models from which the user wishes to extract rules and knowledge. In other words, the DSS does not include step 1 of the EURECA algorithm. This design decision enables the user to select a preferred tool for training the ANN models. After the first step of the EURECA algorithm is completed, then the EURECA DSS supports steps 2 and 3 of the EURECA algorithm, and the MLR equations are generated from the input ANN model (see Figure 6.1). As shown in the upper half of Figure 6.2, the EURECA DSS consists of six modules, which include (1) the graphical user interface (GUI), (2) data input/output (I/O) module, (3) the EURECA modeling module, (4) the rules and data evaluation module, (5) the back calculation module, and (6) the prediction module. Each of these modules and their detailed input and output, except the back calculation module, are shown in Figure 6.3, and will be discussed as follows.
Figure 6.3 – Main components of the EURECA DSS Modules
Each of the modules, except the back calculation module, is described as follows.

6.2.1 Graphical User Interface

The graphical user interface (GUI) of the EURECA DSS supports the functions of (i) data input and output, (ii) EURECA modeling, (iii) rules and data evaluation, (iv) back calculation, and (v) the prediction. The GUI was implemented using the Shiny package (Trademark of RStudio) in R (Chang et al., 2016), which supports researchers to extend their algorithm and analysis to interactive web applications.

6.2.2 Data Input/Output Module

To generate the EURECA model, this module requires as input information related to the studied problem and the trained ANN model, specifically:

- The values of the input or predictor attributes, $x_i$,
- The values of the output or predicted attributes, $y_k$,
- The values of the predicted attributes given by the trained ANN model,
- The weights of the hidden neurons of the trained ANN model, $w_{ij}$, and
- The weights of the output neurons of the trained ANN model, $w_{jk}$.

This module provides the user with two choices: (i) the trained ANN model can be trained with normalized or real values and (ii) the output rules can be in real or normalized value. Based on the user’s selection, the system requires either the dataset in normalized or real values, or both sets of values to generate the EURECA model. The user can upload the studied dataset in the form of CSV files, with data separated either by comma, semicolon or tab. The hidden weights and output weights should be in a matrix.
format as shown in Figure 6.4. Each row in the matrix represents the weighted values between a predictor variable and each of the hidden nodes. For example, the first row in Figure 6.4 represents the weighted values between predictor variable #1 and each of the six hidden nodes labeled by the column headings, and the second row represents the weighted values between the predictor variable #2 and each of the nodes labeled by the column headings and so on.
Figure 6.4 – Hidden weight values in matrix format required by Data I/O Module

<table>
<thead>
<tr>
<th>Node.1</th>
<th>Node.2</th>
<th>Node.3</th>
<th>Node.4</th>
<th>Node.5</th>
<th>Node.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.502</td>
<td>-0.342</td>
<td>0.112</td>
<td>-0.147</td>
<td>0.437</td>
<td>-2.233</td>
</tr>
<tr>
<td>1.49</td>
<td>1.183</td>
<td>-2.712</td>
<td>0.605</td>
<td>-2.457</td>
<td>-1.078</td>
</tr>
<tr>
<td>-0.218</td>
<td>-0.297</td>
<td>-1.067</td>
<td>-1.038</td>
<td>3.397</td>
<td>3.419</td>
</tr>
<tr>
<td>-0.357</td>
<td>0.166</td>
<td>-1.395</td>
<td>0.415</td>
<td>0.036</td>
<td>-3.064</td>
</tr>
<tr>
<td>-6.979</td>
<td>-9.941</td>
<td>-4.073</td>
<td>0.179</td>
<td>4.9</td>
<td>3.634</td>
</tr>
<tr>
<td>-1.948</td>
<td>3.361</td>
<td>-4.094</td>
<td>6.644</td>
<td>-1.311</td>
<td>-0.432</td>
</tr>
<tr>
<td>3.919</td>
<td>2.235</td>
<td>-2.066</td>
<td>1.601</td>
<td>-1.011</td>
<td>2.309</td>
</tr>
<tr>
<td>-0.515</td>
<td>-0.562</td>
<td>0.043</td>
<td>-1.279</td>
<td>-0.285</td>
<td>7.927</td>
</tr>
<tr>
<td>-2.767</td>
<td>-12.356</td>
<td>-1.244</td>
<td>-7.923</td>
<td>-0.356</td>
<td>-4.811</td>
</tr>
</tbody>
</table>
6.2.3 EURECA Modeling Module

In this module, the system generates the EURECA model based on the locations of breakpoints (BPs) specified by the user. The number of breakpoints is defined by the number of values separated by commas and set in parentheses; by default, there are two breakpoints located at (-2.49, 2.49). When the user wishes to define more than two breakpoints, they must be set in sequence from lowest to highest. The module can support any number of breakpoints defined by the user. If both datasets are provided, the module also supports generating rules using either normalized or real values. In addition, the module returns statistics on the accuracy of the generated EURECA model in terms of (i) MSE, (ii) $R^2$, (iii) the plot of the ANN predicted values against the actual values, and (iv) the plot of the EURECA predicted values against the actual values. The user can adjust the location and number of breakpoints according to the statistics displayed until a model with satisfactory accuracy is obtained.

6.2.4 Rules and Data Evaluation Module

This module enables the user to evaluate the results of the generated EURECA model in terms of accuracy and significance of the rule set as a whole and each individual rule, assess the ranking of the significances of the input attributes, and examine each individual rule and its coverage. The module displays the ANVOA table, which shows measure of the rule set’s accuracy and significance. The module also analyzes and displays the ranking and the coverage of each input attribute to the user in a table. A summary of each individual rule is also displayed, which is shown in Figure 6.5. The
summary shows the following information: (1) the coverage of the selected rule, (2) the distribution of the residuals given by the rule, (3) the coefficient and significance of each attribute in the selected rule, and (4) the accuracy and significance of the selected rule. The values under the column “Estimate” are the coefficients of the attributes in the selected rule. The t value and Pr(>|t|) reflect the significances of the attributes. A large t value and a value of Pr(>|t|) less than 0.05 means the attribute is significant. The table also indicates the significance of each input attribute by the number of asterisks (*) next to the attribute. Three asterisks means that the attribute is highly significant with Pr(>|t|) < 0, whereas no asterisks means the value of Pr(>|t|) is greater than 1 and the attribute is considered insignificant. The accuracy of the generated rule is measured by the residual standard error and R², and the significance of the equation is measured by the F-statistic (same as the F ratio explained earlier) and the p-value.
Figure 6.5 – Summary of each individual rule given by the Rules and Data Evaluation Module
6.2.5 Prediction Module

This module supports the user to upload a new input dataset and the module predicts the output values using the EURECA model generated previously by the DSS. If the underlying ANN model was trained by normalized data and the rules were extracted with real data, then the user has to provide both normalized and real values for the new dataset for this prediction module to function.

6.3. Application of the EURECA DSS

The application of the EURECA DSS is demonstrated using a multi-phase flow pipeline dataset. In transportation pipeline, the accurate determination of pressure drop along the pipeline is essential to ensure efficient pipe system design. Therefore, the main objective of the model is to accurately predict the inlet pressure given the outlet pressure and other attributes and the pressure drop is given by the difference between the inlet and outlet pressure. The other objective is to gain understanding of the significance of the predictor attributes on the predicted attribute. The studied dataset was obtained from a subsea pipeline of Bohai Oilfield, China in the winter of 2011 and the properties of the studied pipeline are summarized in Table 6.2. The pipeline transport crude oil, water, and natural gas. In most of the cases, the flow pattern in the pipeline is stratified flow. The properties of the fluid are summarized in Table 6.3.
Table 6.2 – Properties of the Bohai Oilfield pipeline

<table>
<thead>
<tr>
<th>Properties</th>
<th>Measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inner Diameter [m]</td>
<td>0.2017</td>
</tr>
<tr>
<td>Thickness [m]</td>
<td>0.0087</td>
</tr>
<tr>
<td>Length [m]</td>
<td>40.3</td>
</tr>
<tr>
<td>Elevation [m]</td>
<td>-40</td>
</tr>
<tr>
<td>Roughness [m]</td>
<td>4.60E-0.5</td>
</tr>
</tbody>
</table>

Table 6.3 – Properties of fluid and gas in the Bohai Oilfield pipeline

<table>
<thead>
<tr>
<th>Properties</th>
<th>Measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gas density (at standard condition) [kg/m$^3$]</td>
<td>0.7149</td>
</tr>
<tr>
<td>Oil density (at 20 °C) [kg/m$^3$]</td>
<td>904.3</td>
</tr>
<tr>
<td>Oil density (at 50 °C) [kg/m$^3$]</td>
<td>884.8</td>
</tr>
<tr>
<td>Oil API (at 60 °F)</td>
<td>24.08</td>
</tr>
<tr>
<td>Oil viscosity (at 50 °C) [mPa·s]</td>
<td>15.85</td>
</tr>
</tbody>
</table>
The dataset consists of 722 data tuple with six input or predictor attributes including (1) the outlet pressure, (2) inlet temperature, (3) outlet temperature, (4) watercut, (5) gas-oil ratio (GOR) at standard condition (20°C, 1atm), and (6) total mass flow rate. The output or predicted attribute is the inlet pressure of the pipeline. The statistical properties of the predictor and predicted attributes are summarized in Table 6.4. 70% of the dataset was used for training, and 30% of the dataset was used for testing. The attribute values were first normalized to the range of [-1, 1] before feeding into the ANN algorithm for training. An ANN model with five hidden neurons was trained using Rapidminer. The predictive accuracy of the trained ANN model is summarized in Table 6.5. The MSE values were given by the normalized values.
### Table 6.4 – Statistical properties of the predictor and predicted attributes

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Mean</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Predictor Attribute</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 Outlet Pressure, P\textsubscript{out} [bar]</td>
<td>59.44</td>
<td>61.23</td>
<td>60.20</td>
<td>0.2582</td>
</tr>
<tr>
<td>2 Inlet Temperature, T\textsubscript{in} [°C]</td>
<td>39.39</td>
<td>57.01</td>
<td>53.31</td>
<td>1.3959</td>
</tr>
<tr>
<td>3 Outlet Temperature, T\textsubscript{out} [°C]</td>
<td>6.20</td>
<td>12.10</td>
<td>9.91</td>
<td>0.9395</td>
</tr>
<tr>
<td>4 Watercut [%]</td>
<td>0.00</td>
<td>0.16</td>
<td>0.04</td>
<td>0.0171</td>
</tr>
<tr>
<td>5 Gas-oil ratio, GOR [vol/vol]</td>
<td>27.17</td>
<td>300.00</td>
<td>77.22</td>
<td>49.1468</td>
</tr>
<tr>
<td>6 Total mass flow rate, [kg/h]</td>
<td>2773.46</td>
<td>9704.93</td>
<td>5919.55</td>
<td>1545.7160</td>
</tr>
<tr>
<td><strong>Predicted Attribute</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 Inlet Pressure, P\textsubscript{in} [bar]</td>
<td>61.54</td>
<td>63.87</td>
<td>62.75</td>
<td>0.2768</td>
</tr>
</tbody>
</table>

### Table 6.5 – Predictive accuracy of the trained ANN model

<table>
<thead>
<tr>
<th></th>
<th>Training</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE</td>
<td>0.0064</td>
<td>0.0067</td>
</tr>
<tr>
<td>R\textsuperscript{2}</td>
<td>0.8872</td>
<td>0.8794</td>
</tr>
</tbody>
</table>
Both the real and normalized values of the training dataset were uploaded to the EURECA DSS via the Data I/O module, as shown in Figure 6.6.

The EURECA model was generated in the EURECA Modeling module with the locations of breakpoints at (-1.1, 1.1) using the real values of the attributes. The generated EURECA model has an MSE value of 0.0083 (given by real values) and an $R^2$ value of 0.8924, which is slightly more accurate than the trained ANN model. A plot of the predicted output values given by the generated EURECA model and the actual output values returned by the EURECA Modeling module is shown in Figure 6.7.
Figure 6.6 – Both real and normalized values of the dataset uploaded to the EURECA DSS system via the Data I/O module.
Figure 6.7 – Plot of predicted output values given by the generated EURECA model Vs Actual output values given by the EURECA modeling module
The ANOVA table for the generated model and the ranking of the predictor attributes given by the Rules and Data Evaluation module is shown in Figure 6.8. As shown in Figure 6.8, two rule or equations were extracted from the trained ANN model. The F ratio of the rule set is much greater than 1, indicating that there is a strong relationship between the predictor and predicted attributes given by the rules. The P value of the rule set is equal to zero, suggesting the predicted values given by the rules are likely to be as significant as that indicated by the F ratio. The predictor attribute, outlet pressure, was identified as the most dominating input attributes in all of the rules.

The two rules extracted from the pipeline dataset are as follows and the summaries of the two rules given by the Rule and Data Evaluation module are shown in Figure 6.9 and Figure 6.10, respectively:

(1) Rule #1:

IF ((60.05 bar ≤ P_{out} ≤ 60.80 bar) AND (47.31 ℃ ≤ T_{in} ≤ 57.01 ℃) AND (8.2 ℃ ≤ T_{out} ≤ 11.3 ℃) AND (0 % ≤ watercut ≤ 0.0935 %) AND (27.17 ≤ GOR ≤ 189.91) AND (3329.04 kg/hr ≤ mass flow rate ≤ 9700.28 kg/hr))

THEN

P_{in} = 13.46 + 0.8107 P_{out} - 0.0005 T_{in} + 0.0423 T_{out} + 1.769 watercut + 1.00E-04 GOR + 7.38E-06 mass flow rate

The summary of Rule #1 in Figure 6.9 shows that the rule covers 251 of the data tuples. The rule has an R^2 value of 0.8725, and both the values of F-ratio and P-value indicated
that the rule or MLR equation is significant. However, statistically, the input attributes of inlet temperature, GOR, and mass flow rate are considered as insignificant in the MLR equation for this cluster of the dataset.

(2) Rule #2:

IF ((59.44 bar ≤ P_{out} ≤ 60.38 bar) AND (39.39 °C ≤ T_{in} ≤ 55.91 °C) AND (6.2 °C ≤ T_{out} ≤ 11.2 °C) AND (0 % ≤ watercut ≤ 0.1600 %) AND (27.28 ≤ GOR ≤ 300.00) AND (2773.46 kg/hr ≤ mass flow rate ≤ 9704.93 kg/hr))

THEN

P_{in} = -17.82 + 1.3148 P_{out} + 0.0161 T_{in} + 0.0721 T_{out} + 0.6347 watercut – 5.00E-04 GOR – 7.37E-06 mass flow rate

The summary of Rule #2 in Figure 6.10 shows that the rule covers 254 of the data tuples. The rule has an R^2 value of 0.7165, and both the values of F-ratio and P-value indicated that the rule or MLR equation is significant. Statistically, all the input attributes are considered as significant in this cluster of the dataset, except mass flow rate, which has no asterisk given in the summary.
Figure 6.8 – Accuracy, significance and attribute ranking of the generated EURECA model given by the Rules and Data Evaluation Module
Figure 6.9 – Summary of Rule #1 given by the Rule and Data Evaluation Module
### Summary of the selected rule

Total number of data: 254

Call:
`lm(formula = thisFormula, data = cbind(thisRuleInput, thisRuleActualOut))`

Residuals:
```
     Min  1Q Median  3Q     Max
-0.75945 -0.04574  0.01085  0.05222  0.21310
```

Coefficients:
```
                  Estimate  Std. Error   t value  Pr(>|t|)
(Intercept)  -1.782e+01   3.329e+00 -5.353   1.97e-07 ***
Pout         1.315e+00   5.415e-02  24.280 < 2e-16 ***
Tin          1.610e-02   4.102e-03   3.925   0.000112 ***
Tout         7.205e-02   9.349e-03   7.707   3.14e-13 ***
watercut   6.347e-01   3.186e-01   1.992   0.047463 *
GOR         -5.468e-04   1.801e-04  -3.037   0.002647 **
mass        -7.365e-06   5.730e-06  -1.286   0.199820
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Residual standard error: 0.1063 on 247 degrees of freedom
Multiple R-squared: 0.7165,  Adjusted R-squared: 0.7096
F-statistic: 104 on 6 and 247 DF,  p-value: < 2.2e-16

---

**Figure 6.10 – Summary of Rule #2 given by the Rule and Data Evaluation Module**
The testing dataset used to test the trained ANN model was uploaded to the prediction module of the EURECA DSS for prediction as shown in Figure 6.11.

The predicted values given by the EURECA model has an MSE value of 0.0090 and an $R^2$ value of 0.8815, which is slightly more accurate than that given by the trained ANN model as shown in Table 6.5.
Figure 6.11 – New input data uploaded for prediction in the prediction module of the 

EURECA DSS
6.4 Conclusion and Future Work

In this chapter, the development and application of the EURECA DSS were presented. The EURECA DSS uses the EURECA algorithm to extract rules in the form of multiple linear regression equations from trained ANN models. The EURECA algorithm generates rules by clustering the dataset according to the weighted input values and then perform multiple linear regression on each of the clustered data. The DSS allows the user to examine the accuracy of the generated EURECA model through the value of MSE and $R^2$ and the user can adjust the number and locations of the breakpoints until the satisfactory accuracy of the EURECA model is achieved. The significance of the generated model can be examined through the ANOVA table presented by the system. The DSS provides some insight to the user on the relationship between the predictor and predicted attributes by examining the coefficients of the attributes in the extracted rules. Although not discussed in this paper, the DSS also solves the back calculation problems for the user and suggests the likely value of the unknown predictor attributes that will achieve the target predicted value. Last but not least, the DSS supports predictions using the generated EURECA model.

A multi-phase flow pipeline dataset collected from a subsea pipeline of Bohai Oilfield was used to illustrate the application of the developed EURECA DSS. The example demonstrated how to upload the relevant information to generate the EURECA model, how to evaluate the generated model to ensure an accuracy and significant model, and how to examine the results in order to gain insight into the studied problem. The results
given by the DSS showed that the generated EURECA model gave slightly more accurate prediction than the trained ANN model. Only two rules were given for the studied dataset and based on the evaluation of the parameter coefficients, outlet pressure was considered as the most significant attribute on the inlet pressure of the pipeline.

Some directions for future work include:

1. Include modules that support visualization, such as 2D- and 3D-plots of the studied dataset.
CHAPTER 7 CASE STUDY #1 MULTIPHASE FLOW PIPELINE – OPENING THE BLACK BOX OF THE ANN MODEL

In this chapter, the pressure gradient in a two-phase flow pipeline was studied. The objectives of applying the EURECA algorithm on the studied problem are:

1. To provide better understanding on the relationship between the predictor attributes and the pressure gradient along the multiphase flow pipeline;
2. To shed light on the relationships among the data tuples that are assigned to a similar range of weighted input values.

7.1 Background

Two phase flow involving liquid and gas, also called multiphase flow, occurs in almost all pipelines in oil and gas production and transportation. With more than one phase involved, accurately predicting the behavior of the fluid becomes more challenging. The accurate determination of pressure drop along the pipeline is essential when designing an efficient pipe system and selecting an optimum production strategy. Numerous correlation models have been developed since the early 1950s (Jerez-Carrizales et al., 2015). Some of the correlations most commonly used by the industry (Jahanandish et al., 2011) include: (Hagedorn & Brown, 1961), (Duns & Ros, 1963), (Orkiszewski, 1967), and (Mukherjee & Brill, 1985). Among which, the Beggs-Brill method was found to be most successful in stratified flow (Spedding et al., 2006) and it can be applied to pipelines in all directions. However, the main problems with empirical correlations are that (1) they are usually based on small scale experiment and may not be as accurate
when scaled up to industrial size systems, and (2) to improve correlation accuracy, a large amount of data is required, which could be difficult to obtained.

To overcome the drawbacks of empirical correlations, researchers turn to machine learning algorithms, such as artificial neural network (ANN), support vector machines (SVM) and adaptive neuro-fuzzy inference system (ANFIS). Alizadehdakhel et al (2009) trained an ANN model to predict the pressure drop along horizontal and inclined pipelines that carry air and water using experimental datasets. Ozbayoglu & Ozbayoglu (2009) applied the ANN algorithm to estimate the flow pattern and pressure drop along horizontal annular geometries. Their results showed less than 30% error in the pressure drop prediction. El-Sebakhy (2010) used neuron-fuzzy systems to identify flow regimes and estimate holdup in horizontal pipes with two-phase gas-liquid flow. Their results showed that the neuro-fuzzy models outperformed both conventional empirical correlations and neural network models. Instead of predicting pressure change along the pipeline, Bar et al. (2010) developed ANN models to determine the pressure gradient through different pipe components, such as elbow, orifice, gate valve and globe valve. Chakrabarti et al. (2010) trained an ANN model using over 500 experimental data points and with predictive accuracy of less than 4% from the experimental results. Ayoub & Elraies (2014) employed the group method of data handling-type neural network to estimate the pressure drop of gas-liquid flow in inclined pipelines. Fazavi et al. (2014) employed the least square support vector machine algorithm to estimate the pressure drop in oil-water horizontal pipelines. The model was generated using over 700 experimental data points from literatures. Cozi et al. (2016) employed the ANN algorithm to predict
characteristics of slug flows in horizontal pipes. Halali et al. (2016) developed a radial basis function neural network model using over 900 experimental data points from literature to estimate the pressure gradient in water-oil horizontal pipes. Their results showed that the generated model outperformed the correlation approaches.

7.2 The Dataset

The dataset describes the pressure gradient of the two-phase water-oil flow in pipelines. The dataset consists of 376 data tuples that were extracted from the experimental results of the following papers: (Liu et al., 2009), (Al-Yaari et al. 2009), (Al-Wahaibi et al., 2007), (Lovick & Angeli, 2004) and (Elseth, 2001). The dataset has 5 numerical predictor or input attributes: (1) oil superficial velocity (\( V_{\text{oil}} \)), (2) water superficial velocity (\( V_{\text{water}} \)), (3) pipe diameter (\( P_d \)), (4) pipe roughness (\( P_{\text{rough}} \)) and (5) oil viscosity (\( V_{\text{is}}_{\text{oil}} \)), and one predicted or output attribute: the pressure gradient along the pipeline (dp/dx). Table 7.1 gives the summary of the predictor and predicted attributes of the dataset.

**Figure 7.1** shows the 3D plot of \( \frac{dp}{dx} \) Vs \( V_{\text{oil}} \) Vs \( V_{\text{water}} \). It can be observed that the dataset was segmented into five portions depending on the pipe diameter. Based on this observation, it was hypothesized that the pipeline diameter is one of the key factors affecting the values of the weights between the input parameters and the hidden neurons during ANN modeling. Hence, the hypothesis was formed that the pipeline diameter was directly correlated to how the data tuples were clustered in the EURECA algorithm.
Table 7.1 – Summary of predictor and predicted attributes of the water-oil pipeline dataset

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Oil superficial velocity, (V_{oil}) [m/s]</td>
<td>0.10 - 2.73</td>
<td>0.05 - 0.63</td>
<td>0.09 - 0.44</td>
<td>0.00 - 3.00</td>
<td>0.00 - 3.35</td>
</tr>
<tr>
<td>Water superficial velocity, (V_{water}) [m/s]</td>
<td>0.1 - 2.8</td>
<td>0.065 - 0.4</td>
<td>0.1 - 0.8</td>
<td>0.0 - 3.0</td>
<td>0.0 - 3.35</td>
</tr>
<tr>
<td>Pipe Diameter, (P_d) [mm]</td>
<td>25.4</td>
<td>26.1</td>
<td>14.0</td>
<td>38.0</td>
<td>56.3</td>
</tr>
<tr>
<td>Pipe roughness, (P_{rough}) [m]</td>
<td>1.00E-05</td>
<td>1.00E-05</td>
<td>1.00E-05</td>
<td>4.50E-05</td>
<td>1.00E-05</td>
</tr>
<tr>
<td>Oil viscosity, (Vis_{oil}) [mPa-s]</td>
<td>1.57</td>
<td>3.47</td>
<td>5.50</td>
<td>6.00</td>
<td>1.64</td>
</tr>
<tr>
<td>Pressure Gradient, (dp/dx) [Pa/m]</td>
<td>28.62 – 1007.89</td>
<td>33.10 – 749.37</td>
<td>110.87 – 1606.30</td>
<td>177.00 – 2559.00</td>
<td>38.42 – 1307.84</td>
</tr>
</tbody>
</table>
Figure 7.1 – 3D plot of pressure gradient Vs oil superficial velocity Vs water superficial velocity
7.3 Application of the EURECA algorithm

Step 1: Training the ANN models

The ANN model involves a feedforward backpropagation neural network that uses the sigmoid function as the activation function for the hidden neurons and the linear function for the activation function for the output neurons. It was implemented using Rapidminer (trademark of Rapidminer) with the following parameter settings:

- Number of hidden nodes: 4
- Learning rate: 0.01
- Momentum: 0.01
- Training cycles: 5,000
- Validation: 20-fold cross-validation

Rapidminer produced two output items useful for step 2 of the application: (i) the trained ANN model produced, (ii) the weights of the hidden neurons of the trained ANN model.

Step 2: Clustering the dataset according to their weighted input values

First, the EURECA algorithm requires from the user information on the break points. For example, the breakpoints provided are at (-2.49, 2.49). After running the EURECA system with the input from step 1, the output is shown in Figure 7.2. Figure 7.2 shows the first 10 data tuples of the dataset after clustering. The numbers under the column “Node.X” represent the group node X of a particular data tuple. The number “1” represents group A where the weighted input value is less than -2.49, number “2”
represents group B where the weighted input value is between -2.49 and 2.49, and number “3” represents group C where the weighted input value is greater than 2.49. As shown in Figure 7.2, data tuple #1 had a hidden neuron group combination of (group B, group B, group C, group B) and it was assigned to cluster or Rule #1. All the data tuples that had the same hidden neuron group combination were also assigned to Rule #1. Similarly, data tuple #7 had a hidden neuron group combination of (group B, group B, group B, group B) and it was assigned to cluster or Rule #2, and all the data tuples that had the same combination were assigned to Rule #2. A total of 14 clusters or rules with different hidden neuron group combination were generated.

The data tuples were clustered according to the normalized values, and after they were clustered, they were mapped back to their original values by the following equation:

\[
\text{Normalized value} = \frac{(\text{actual value} - \min(data)) \times (b - a)}{\max(data)} + \min(data)
\]

where

- \([a, b]\) is the target range of values that we want to normalize to, in this case, \(a = -1\) and \(b = 1\);
- \(\min(data)\) is the minimum value of the given attribute, for example, if we are normalizing attribute \(X1\), then \(\min(data)\) is the minimum value of the attribute \(X1\);
- \(\max(data)\) is the maximum value of the given attribute.
<table>
<thead>
<tr>
<th>Vo</th>
<th>Vw</th>
<th>Pdia</th>
<th>Prough</th>
<th>OilVis</th>
<th>PG</th>
<th>Node.1</th>
<th>Node.2</th>
<th>Node.3</th>
<th>Node.4</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.09</td>
<td>0</td>
<td>25.4</td>
<td>1.00E-05</td>
<td>1.57</td>
<td>28.62</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>0.09</td>
<td>0.05249</td>
<td>25.4</td>
<td>1.00E-05</td>
<td>1.57</td>
<td>36.78</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>0.09</td>
<td>0.164215</td>
<td>25.4</td>
<td>1.00E-05</td>
<td>1.57</td>
<td>69.44</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>0.09</td>
<td>0.27517</td>
<td>25.4</td>
<td>1.00E-05</td>
<td>1.57</td>
<td>103.72</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>0.09</td>
<td>0.38720</td>
<td>25.4</td>
<td>1.00E-05</td>
<td>1.57</td>
<td>152.12</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>0.09</td>
<td>0.4418</td>
<td>25.4</td>
<td>1.00E-05</td>
<td>1.57</td>
<td>177.36</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>0.09</td>
<td>0.552735</td>
<td>25.4</td>
<td>1.00E-05</td>
<td>1.57</td>
<td>247.03</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>0.09</td>
<td>0.66443</td>
<td>25.4</td>
<td>1.00E-05</td>
<td>1.57</td>
<td>315.155</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>0.09</td>
<td>0.77574</td>
<td>25.4</td>
<td>1.00E-05</td>
<td>1.57</td>
<td>377.255</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>0.09</td>
<td>0.942675</td>
<td>25.4</td>
<td>1.00E-05</td>
<td>1.57</td>
<td>503.855</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

*Figure 7.2 – First 10 data tuples clustered according to their weighted input values*
Step 3: Perform MLR on each of the clusters

For each cluster of data, the linear regression function, \textit{lm()}, implemented in R, was used to model the multiple linear regression equations by finding the fitted line using the least squares deviation method. A summary of Rule #1 given by the \textit{lm()} function is shown in Figure 7.3.

The summary of the equations shows the values (labeled as A) and the significance (labeled as B) of the estimated coefficients, as well as the error and significance of the estimated function (labeled as C). The values under the column “Estimate” in A are the coefficients of the attributes in the MLR equation. Based on the values given in A, Rule #1 can be expressed as follows:

\[
\frac{dP}{dx} = -119.13 + 1422.55 V_{oil} + 389.61 V_{water}
\]
Figure 7.3 – Summary of Rule #1 given by the `lm()` function in R
The t value and Pr(>|t|) in B reflect the significance of the attributes. A large t value and a value of Pr(>|t|) less than 0.05 mean the attribute is significant. R also indicates the significances using the number of asterisks or “*” beside the attribute. Three “*” means that the attribute is highly significant with Pr(>|t|) < 0; whereas no asterisk indicates the value of Pr(>|t|) is greater than 1 and the attribute is considered insignificant. In the example shown in Figure 7.3, three of the coefficients are not defined because of singularity; this means the values of those three attributes, namely pipe diameter (P\text{dia}), pipe roughness (P\text{rough}) and oil viscosity (OilVis) have little or no variations in the given cluster and the coefficients of those attributes cannot be resolved. The values of multiple $R^2$ and adjusted $R^2$ given in C reflect the accuracy of the derived MLR equation, and the closer the value of the multiple and adjusted $R^2$ to 1, the more accurate is the equation. Note that the adjusted $R^2$ is a modified version of the multiple $R^2$. The adjusted $R^2$ takes into account the number of predictor variables and the value will be decreased if insignificant predictor variables are included. The values of F-statistic and p-value measure the significance of the equation and a value of greater than 1 and less than 0.05, respectively, is ideal for significance. Some of the outputs also given by the \texttt{lm()} function, which are not shown here, include the fitted values – the predicted output values given by the generated linear regression function, and the residual values – differences between the predicted and actual output.

As shown in Figure 7.3, Rule #1 had $R^2$ of 0.9905 which reflects the high accuracy of the generated rule. The predictor attributes oil superficial velocity, $V_{\text{oil}}$, and water superficial
velocity, $V_{water}$, were both considered as significant. No coefficients were found for the predictor attributes $P_{dia}$, $P_{rough}$ and $Vis_{oil}$ due to singularity, i.e. the values of these attributes had no or little variation in the given cluster.

7.4 Results and Analysis

7.4.1 Accuracy of the EURECA model

The accuracies of the trained ANN model and the generated EURECA model in terms of MSE and $R^2$ are summarized in Table 7.2. As shown in Table 7.2, the predictive accuracy of the EURECA model is slightly better than that of the trained ANN model with a lower MSE value and higher $R^2$ value. The MSE values given by the ANN and EURECA models are 13616.16 and 13422.54, respectively. The $R^2$ value given by the ANN and EURECA models are 0.9437 and 0.9445, respectively. Therefore, the generated EURECA model is considered to have satisfactory accuracy.
Table 7.2 – Comparing Accuracies of the ANN model and the EURECA model for the water-oil pipeline dataset

<table>
<thead>
<tr>
<th></th>
<th>ANN Model</th>
<th>EURECA Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R^2$</td>
<td>0.9437</td>
<td>0.9445</td>
</tr>
<tr>
<td>MSE</td>
<td>13616.16</td>
<td>13422.54</td>
</tr>
</tbody>
</table>
7.4.2 Knowledge Extracted from the EURECA Model

A total of 14 rules were given by the EURECA model. Each of these rules covers a specific portion of the problem space in the studied problem. The coverage of each rule reflects the significance of the rule in the given problem domain. The most significant rule extracted for this dataset is Rule #11 which states that

\[
\text{IF } \quad \begin{align*}
(P_d = 56.3 \text{ mm}) & \quad \text{AND} \quad (0.67 \text{ m/s} \leq V_{\text{oil}} < 2 \text{ m/s}) \\
& \quad \text{AND} \quad (0 \leq V_{\text{water}} < 1.002 \text{ m/s})
\end{align*}
\]

\[
\text{THEN } \quad \frac{dP}{dx} = 335.06 V_{\text{oil}} + 86.40 V_{\text{water}} - 185.33
\]

From the above rule, it can be observed that oil superficial velocity has the largest coefficient and hence is the most dominating attribute in the equation. The complete set of rules along with the coverage of each rule is shown in Table 7.3. By examining the attributes’ coefficients in all the equations, it can be seen that the oil superficial velocity is the most significant predictor attribute because in 12 out of the 14 equations extracted, oil superficial velocity has the largest coefficient. This observation agrees with that made in (Halali et al., 2016) in which the authors trained a radial basis function neural network model and applied sensitivity analysis on the trained model. The sensitivity analysis revealed the relative importance of the predictor attributes on pressure gradient and the results showed that oil superficial velocity has the most significant impact on pressure drop along the pipeline, followed by water superficial velocity.
Table 7.3 – Generated EURECA rules for the water-oil pipeline dataset

<table>
<thead>
<tr>
<th>Rule</th>
<th>Coverage</th>
<th>IF...</th>
<th>THEN ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>13</td>
<td>( P_d = 25.4 \text{ mm} ) AND ( 0.09 \text{ m/s} \leq V_{\text{oil}} &lt; 0.23 \text{ m/s} ) AND ( 0 \text{ m/s} \leq V_{\text{water}} &lt; 0.442 \text{ m/s} )</td>
<td>( \frac{dP}{dx} = 1422.55 V_{\text{oil}} + 389.61 V_{\text{water}} - 119.13 )</td>
</tr>
<tr>
<td>#2</td>
<td>43</td>
<td>( 14 \text{ mm} \leq P_d &lt; 26.1 \text{ mm} ) AND ( 0.09 \text{ m/s} \leq V_{\text{oil}} &lt; 0.63 \text{ m/s} ) AND ( 0.099 \text{ m/s} \leq V_{\text{water}} &lt; 0.997 \text{ m/s} ) AND ( 1.00 \times 10^{-5} \text{ m} \leq P_{\text{rough}} &lt; 4.5 \times 10^{-5} \text{ m} ) AND ( 1.57 \text{ mPa-s} \leq V_{\text{viscosity}} &lt; 5.5 \text{ mPa-s} )</td>
<td>( \frac{dP}{dx} = 1780.50 V_{\text{oil}} + 1092.29 V_{\text{water}} - 27.35 P_{\text{dia}} - 8860937 P_{\text{rough}} )</td>
</tr>
<tr>
<td>#3</td>
<td>3</td>
<td>( P_d = 26.1 \text{ mm} ) AND ( V_{\text{oil}} = 0.05 \text{ m/s} ) AND ( 0.125 \text{ m/s} \leq V_{\text{water}} &lt; 0.312 \text{ m/s} )</td>
<td>( \frac{dP}{dx} = 435.26 V_{\text{water}} - 14.48 )</td>
</tr>
<tr>
<td>#4</td>
<td>15</td>
<td>( P_d = 26.1 \text{ mm} ) AND ( 0.21 \text{ m/s} \leq V_{\text{oil}} &lt; 0.63 \text{ m/s} ) AND ( 0.066 \text{ m/s} \leq V_{\text{water}} &lt; 0.514 \text{ m/s} )</td>
<td>( \frac{dP}{dx} = 932.36 V_{\text{oil}} + 669.89 V_{\text{water}} - 199.42 )</td>
</tr>
<tr>
<td>#5</td>
<td>2</td>
<td>( P_d = 38 \text{ mm} ) AND ( 0.8 \text{ m/s} \leq V_{\text{oil}} &lt; 1.0 \text{ m/s} ) AND ( V_{\text{water}} = 0.0 \text{ m/s} )</td>
<td>( \frac{dP}{dx} = 735 V_{\text{oil}} - 368 )</td>
</tr>
<tr>
<td>#6</td>
<td>16</td>
<td>( P_d = 38 \text{ mm} ) AND ( 0.8 \text{ m/s} \leq V_{\text{oil}} &lt; 1.0 \text{ m/s} ) AND ( 0.8 \text{ m/s} \leq V_{\text{water}} &lt; 1.0 \text{ m/s} )</td>
<td>( \frac{dP}{dx} = 460.83 V_{\text{oil}} - 178.33 )</td>
</tr>
<tr>
<td>#7</td>
<td>4</td>
<td>( P_d = 38 \text{ mm} ) AND ( V_{\text{oil}} = 0.0 \text{ m/s} ) AND ( 0.8 \text{ m/s} \leq V_{\text{water}} &lt; 3.0 \text{ m/s} )</td>
<td>( \frac{dP}{dx} = 945.50 V_{\text{water}} - 647.41 )</td>
</tr>
<tr>
<td>#8</td>
<td>2</td>
<td>( P_d = 38 \text{ mm} ) AND ( 0.8 \text{ m/s} \leq V_{\text{oil}} &lt; 3.0 \text{ m/s} ) AND ( 0.8 \text{ m/s} \leq V_{\text{water}} &lt; 3.0 \text{ m/s} )</td>
<td>( \frac{dP}{dx} = 1170 V_{\text{oil}} - 1012 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>AND</td>
<td>1.5 m/s ≤ (V_{oil}) &lt; 2.0 m/s AND (V_{water} = 0) m/s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#9</td>
<td>52</td>
<td>(P_d = 38) mm AND 1.5 m/s ≤ (V_{oil}) &lt; 3.0 m/s AND 1.5 m/s ≤ (V_{water}) &lt; 3.0 m/s</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(dP/dx = 864.41 \ V_{oil} - 807.92)</td>
<td></td>
</tr>
<tr>
<td>#10</td>
<td>15</td>
<td>(P_d = 56.3) mm AND 0.4 m/s ≤ (V_{oil}) &lt; 0.67 m/s AND 0.254 ≤ (V_{water}) &lt; 1.001 m/s</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(dP/dx = 299.51 \ V_{oil} - 11.95 \ V_{water} - 37.15)</td>
<td></td>
</tr>
<tr>
<td>#11</td>
<td>62</td>
<td>(P_d = 56.3) mm AND 0.67 m/s ≤ (V_{oil}) &lt; 2.0 m/s AND 0.0 ≤ (V_{water}) &lt; 1.001 m/s</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(dP/dx = 335.06 \ V_{oil} + 86.40 \ V_{water} - 185.33)</td>
<td></td>
</tr>
<tr>
<td>#12</td>
<td>8</td>
<td>(P_d = 56.3) mm AND 1.5 m/s ≤ (V_{oil}) &lt; 2.0 m/s AND 0.670 ≤ (V_{water}) &lt; 0.999 m/s</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(dP/dx = 587.79 \ V_{oil} + 254.88 \ V_{water} - 765.76)</td>
<td></td>
</tr>
<tr>
<td>#13</td>
<td>5</td>
<td>(P_d = 56.3) mm AND 2.0 m/s ≤ (V_{oil}) &lt; 2.5 m/s AND 0.0 ≤ (V_{water}) &lt; 0.192 m/s</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(dP/dx = 480.19 \ V_{oil} - 751.84 \ V_{water} - 359.57)</td>
<td></td>
</tr>
<tr>
<td>#14</td>
<td>23</td>
<td>(P_d = 56.3) mm AND 2.5 m/s ≤ (V_{oil}) &lt; 3.0 m/s AND 0.0 ≤ (V_{water}) &lt; 1.0 m/s</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(dP/dx = 755.09 \ V_{oil} + 294.36 \ V_{water} - 1270.53)</td>
<td></td>
</tr>
</tbody>
</table>
7.5 Discussion: Opening the “Black Box”

As explained earlier, the assumption made when developing the EURECA algorithm is that the data tuples that are assigned to the same range of weighted input values share some characteristics among their predictor attributes and some linear relationship with the predicted attributes. By examining the 3D plot of pressure gradient Vs oil superficial velocity Vs water superficial velocity, the hypothesis made was that there was a correlation between the pipe diameters and the assigned ANN algorithm weights.

It can be observed from Table 7.3 that apart from Rule #2, all the other rules cover only one pipe size. Rule #1 covers data tuples with pipe diameter equals to 25.4 mm only. Rule #3 and #4 cover data tuples with a pipe diameter of 26.1 mm only. Rule #5 – #9 cover data tuples with pipe diameter equal to 38 mm only and Rule #10 - #14 cover data tuples with pipe diameter equals to 56.3 mm only. Since the generated rules are defined by the values of the weighted input, which in turn depend on the assigned weights between the input attributes and the hidden neurons, this observation supports the hypothesis that the pipe diameter is a major factor affecting the way the ANN algorithm assigns the weight values and models the problem space.

Each subset of data that shares the same pipe diameter is further divided into sub-groups of data according to the values of oil and water superficial velocity and each of the sub-group shares a linear relationship between the predictor and predicted attributes. The EURECA algorithm models the fitted curve given by the ANN algorithm with a set of
planes. **Figure 7.4a** shows the curve given by the ANN algorithm that models the subset of data tuples with pipe diameter equals to 56.3 mm. Each of the 5 rules that share this pipe diameter is visualized as planes generated by the EURECA algorithm; these are shown in **Figure 7.4b – 7.4f**.
Figure 7.4a – Curve given by ANN model for data tuples with $P_d = 56.3$ mm;

Figure 7.4b – Plane given by Rule #10
Figure 7.4c – Plane given by Rule #11

Figure 7.4d – Plane given by Rule #12
Figure 7.4e – Plane given by Rule #13

Figure 7.4f – Plane given by Rule #14
The water-oil pipeline example illustrates how the “black-box” of the ANN algorithm was opened regarding the correlation between the assigned weight values and the predictor attributes, as well as the correlation between the assigned weight and the linear relationship that exists among subsets of data in a given problem domain. However, the dataset used is relatively simple. Since the pipe roughness is constant for a given pipeline and each experimental study was only carried out on one type of oil, the three parameters remain constant within one experiment. Therefore, the ANN algorithm viewed the problem as one with only three predictor attributes, namely oil superficial velocity, water superficial velocity and pipe diameter. As a result, the output rules EURECA generated do not include the attributes of pipe roughness and oil viscosity, except Rule #2, which has a combined dataset with three different pipe diameter sizes. In reality, problems that require the ANN algorithm for modeling are usually highly non-linear and more complex than this studied problem and therefore, the discovered relationships may not be mapped out as clearly as in this case study.

7.6 Conclusion

In this chapter, the problem of pressure gradient along two-phase flow pipelines was study. The dataset consists of 376 data tuples that were extracted from the experimental results of 5 studies. The application of the EURECA algorithm in this case study serves two purposes. First, the case illustrated how the extracted rules successfully provide explicit relationships between the predictor and predicted attributes. The EURECA algorithm was able to produce a total of 14 equations with satisfactory predictive
accuracy for the water-oil pipeline dataset. By examining the coefficients of the predictor attributes in the generated equations, it can be seen that oil superficial velocity is the most significant parameter that influences the pressure gradient in a two-phase flow pipeline. The second objective of application case study was to demonstrate how the extracted rules can provide explicit knowledge by opening the “black-box” of the ANN algorithm and therefore enhancing our understanding of both learning process of the ANN algorithm and the studied problem. By examining the conditions of the rules, it was revealed that there is a correlation between the assigned weight values and the predictor attribute of pipe diameter. The observation suggests a linear relationship exists among the subset of data that shares similar characteristics in their predictor attributes.
CHAPTER 8 CASE STUDY #2 CARBON DIOXIDE CAPTURE SYSTEM –
BACK CALCULATION WITH EURECA MODEL

Carbon dioxide (CO₂) is one of the major components in the greenhouse gas and is one of the main contributors to global warming. Since combustion of fossil fuels is responsible for almost 87 percent of all human-generated CO₂ (Le Quéré et al., 2012), the post-combustion CO₂ capture technology has become an essential technology for combating the problem (Svendsen et al., 2011; Merkel et al., 2010). However, the CO₂ capture process system has over 140 operational parameters, and it is difficult to operate the process (Zhou et al., 2013). A good understanding of the key parameters and their influences on the plant performances is essential for optimizing the process operation and maintaining satisfactory plant efficiency. Often during operation, the process operators need to accurately assess or predict the underlying operational trends based on the current or fluctuating values of the critical process parameters. When the efficiency of the CO₂ capture process is not satisfactory, the operators also need to identify and adjust the correct process parameters so as to re-establish the desired system performances in a timely fashion. Often, the operator needs to achieve some target system performance such as a certain CO₂ production rate or CO₂ absorption efficiency, and he needs to determine the values of other process parameters, such as steam flow rate through reboiler, amine circulation rate, so that the target performance can be achieved. In this study, this specific problem is referred to as the problem of “back calculation.”

In this chapter, the EURECA algorithm was applied to solve the operational problems related to the amine-based CO₂ capture pilot plant at the Clean Energy Technology
Research Institute (CETRI) (formerly known as the International Test Center for CO₂ capture (ITC)) located at the University of Regina, Regina, Saskatchewan, Canada. In this study, we assumed two levels of user expertise and suggested both a novice and an expert approach to solve the back calculation problem using these extracted rules or equations.

8.1 Background

8.1.1 Application Problem Domain

The CO₂ capture process system aims to reduce the amount of CO₂ being released into the atmosphere by capturing and removing CO₂ from industrial gas streams. The process of the amine-based CO₂ capture system at the Clean Energy Technology Research Institute (CETRI) is shown in Figure 8.1 and briefly described as follows (Idem et al., 2006, Zhou et al., 2011a). The numbers shown in Figure 1 correspond to the numbers in the parentheses in the description.
Figure 8.1 – Configuration of CO₂ capture plant at CETRI (Zhou et al., 2013)
Before CO₂ is removed from the flue gas, it is pretreated in the inlet gas scrubber for cooling, and the sulfur oxide (SO₃) and nitrogen oxide (NOₓ) in the flue gas are removed (1). Then, the pre-treated flue gas is passed into the absorber (2), which contains amine solution provided by the lean amine storage tank (3) and heated by the high-temperature steam provided by the boiler (4). The lean amine solution in the absorber absorbs CO₂ from the pre-treated flue gas. The CO₂-free flue gas travels to the off-gas scrubber (6) for cooling and then is released to the atmosphere. The CO₂-rich amine solution in the absorber is passed through the rich amine surge vessel (7) and then to the lean/rich amine heat exchanger (8). The CO₂-rich amine is heated to about 105°C in the heat exchanger and then passed to the stripper (9), where CO₂ is extracted from the rich amine solution using the steam provided by the reboiler (10), and the rich amine solution becomes lean amine solution again. Most of the lean amine solution travels to the lean amine cooler (11) for cooling and then is recycled back to the lean amine storage tank for further CO₂ absorption. A small portion of the lean amine is sent to the reclaimer (12), in which degradation by-products are removed. The residual amine travels back to the stripper through the reflux accumulator (14). The wet CO₂ product exits the reclaimer and travels to the reflux condenser (13), in which water is condensed. The CO₂ enters the CO₂ wash scrubber (15) where it is cooled down and then sent to the dryer and purification unit to produce food grade CO₂ (16).

8.1.2 Previous studies on the CO₂ Capture Process System at CETRI

A series of studies were conducted on the CO₂ capture process plant at CETRI with the objective of obtaining a better understanding of the relationships among the key
parameters of the CO$_2$ capture process system. It was assumed that such an understanding would enhance plant efficiency and reduce the cost of operation. Based on an extensive literature review, four key parameters were selected as indicators of evaluating the process efficiency and plant performance (Zhou et al., 2011a), and they include:

1. CO$_2$ production rate: the amount of wet CO$_2$ extracted from the flue gas,
2. Reboiler heat duty: the amount of heat used for solvent regeneration,
3. Lean loading: the amount of CO$_2$ in the regenerated amine solvent, and
4. CO$_2$ absorption efficiency: the amount of CO$_2$ extracted from the flue gas.

The four indicators were considered as the predicted or output parameters in the modeling studies, and the eight parameters that influence the values of the output parameters become the predictor or input parameters, and they include:

1. Mass flow of CO$_2$ in the flue gas
2. CO$_2$ concentration in the flue gas
3. Off gas flow rate;
4. Amine circulation rate;
5. Reboiler pressure;
6. Pressure of steam through reboiler;
7. Steam flow rate through reboiler, and;
8. Amine concentration.

Various modeling algorithms were applied to analyze the vast dataset that was collected from the CO$_2$ capture process system at CETRI between the years 2003 and 2006. A statistical regression analysis using SPSS (Trademark of SPSS Inc.) was first carried out to investigate the correlations between the identified predictor and predicted variables.
(Zhou et al., 2009), and four mathematical models were built. The predicted variables: CO2 production rate, heat duty, and lean loading were found to have a linear relationship with the identified predictor variables whereas CO2 absorption efficiency was found to have a non-linear relationship with the predictor variables. However, the generated mathematical model only had a predictive accuracy of 70% – 80% and the modeling algorithms provided by SPSS cannot deal with the uncertainty associated with the data.

To improve the predictive accuracy and overcome the weaknesses of the statistical regression approach, the artificial neural network (ANN) approach combined with sensitivity analysis (SA) was adopted (Wu & Chan, 2011). Four feedforward neural network models, one for each of the predicted variables, were built using Weka (Trademark of Weka). The trained ANN models had a predictive accuracy of 88% – 99%. Wu & Chan (2011) then applied sensitivity analysis (SA) to the trained ANN models to identify the relatively more significant predictor attributes. By removing the less significant predictor attributes, the refined datasets were generated. The refined datasets were then validated by the domain experts and were used to re-train the ANN models. The refined predictive models had a predictive accuracy of 85% – 99%. The results showed that by removing the insignificant attributes, the predictive accuracies of the generated models were not diminished. However, one major drawback of the ANN approach was that the generated models were opaque and did not shed light on the relationships between the predictor and predicted variables.
To generate rules that can illustrate the relationship between the predictor and predicted variables, the neuro-fuzzy approach was adopted (Zhou et al., 2010). In this study, the adaptive-network-based fuzzy inference system (ANFIS) was applied to the refined dataset developed in (Wu & Chan, 2011). The models generated gave a predictive accuracy between 88% – 97%. However, there were two major drawbacks of the ANFIS approach. First, in order to generate rules, the parameters needed to be subdivided into categories such as high, medium and low. Expert knowledge was required in this process, and this could introduce subjective bias, which adversely affected the generated fuzzy inference system. The second drawback was that the number of rules generated was equal to the number of input subspace, and depending on the number of input attributes, this could potentially generate a large rule set. Each of the developed inference systems for each of the four predicted or output parameters in (Zhou et al., 2010) had 144 rules. The size of the rule sets meant that the rules were impossible for human to comprehend or for the domain expert to validate. Also, the functions generated by the ANFIS approach were not explicit.

To address the drawbacks of the ANFIS approach in modeling the CO₂ capture dataset, an ANN rule extraction algorithm called PWL-ANN was developed (Chan & Chan, 2017). The PWL-ANN algorithm aims to generate explicit rules from the dataset. The approach involves extracting rules from the trained ANN models by approximating the sigmoid functions in the hidden neurons with some piece-wise linear functions. The extracted rules were in the form of multiple linear regression equations, which represent the relationships between the predictor and predicted variables. Four PWL-ANN models
were generated using the original CO₂ capture dataset with no refining, and the number of rules generated ranges from 38 – 58. The generated models had predictive accuracies that were within 3% of the trained ANN models. By analyzing the magnitude of predictor attributes’ coefficients in the generated rules or equations, the predictor attributes could be ranked from most to least significant in terms of their effects on the predicted attributes. However, one of the major drawbacks of the PWL-ANN algorithm was that the attributes in the extracted rules were in terms of the values that were used to train the ANN models. When the underlying ANN models were trained with values there were normalized to the range of [-1, 1] as in (Chan & Chan, 2017a), the values that were in the rules were also in terms of the normalized values and the generated rules do not have physical mean to the domain expert.

In this study, the proposed EURECA algorithm can overcome the drawbacks of both the ANFIS, and the PWL-ANN approach, by generating explicit rules in the form of multiple linear regression equations in terms of the real values of the attributes. However, unlike the previous modeling studies, which objective was to better understand the relationships between the predictor and predicted parameters of the CO₂ capture process, the main objective of this case study was to use the generated rules from the EURECA algorithm to solve the operational problems in the CO₂ capture system.
8.2. Preparation for Solving the Operational Problems

8.2.1 The Dataset

The dataset used in this study was collected from the CO$_2$ process system in CETRI between the years 2003 and 2006. The dataset has a total of 10,420 data tuples. The dataset has been preprocessed, and the data were in Excel files with no missing values and could be used directly for ANN model training and rule extraction. The statistical properties of the predictor and predicted variables are summarized in Table 8.1. For simplicity, the eight predictor variables were labeled as $x_1$ to $x_8$, and the predicted variable in each of the ANN and EURECA model was labeled as $y$. 

---

214
Table 8.1 – Statistical properties of the predictor and predicted variables of the CO₂ process datasets

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Mean</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Predictor Variable</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Absorber in gas actual flow, $x_1$ [tones/day]</td>
<td>0.267</td>
<td>1.041</td>
<td>0.911</td>
</tr>
<tr>
<td>2</td>
<td>CO₂ concentration of flue gas, $x_2$ [%]</td>
<td>3.648</td>
<td>11.977</td>
<td>10.999</td>
</tr>
<tr>
<td>3</td>
<td>Off gas flow rate, $x_3$ [kg/min]</td>
<td>3.262</td>
<td>4.447</td>
<td>3.807</td>
</tr>
<tr>
<td>4</td>
<td>Amine circulation rate, $x_4$ [kg/min]</td>
<td>7.888</td>
<td>15.116</td>
<td>8.924</td>
</tr>
<tr>
<td>5</td>
<td>Reboiler pressure, $x_5$ [kPa]</td>
<td>1.039</td>
<td>97.091</td>
<td>20.956</td>
</tr>
<tr>
<td>6</td>
<td>Steam pressure of reboiler, $x_6$ [kPa]</td>
<td>104.522</td>
<td>510.752</td>
<td>396.912</td>
</tr>
<tr>
<td>7</td>
<td>Steam flow rate of reboiler, $x_7$ [kg/hr]</td>
<td>21.156</td>
<td>79.979</td>
<td>60.428</td>
</tr>
<tr>
<td>8</td>
<td>Amine concentration, $x_8$ [molarity]</td>
<td>2.910</td>
<td>8.025</td>
<td>5.468</td>
</tr>
<tr>
<td><strong>Predicted Variable, y</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>CO₂ production rate [tones/day]</td>
<td>0.197</td>
<td>0.999</td>
<td>0.671</td>
</tr>
<tr>
<td>11</td>
<td>Heat duty [BTU/lb-mole CO₂]</td>
<td>35604.6</td>
<td>120986.0</td>
<td>91881.4</td>
</tr>
<tr>
<td>12</td>
<td>Lean loading [dimensionless]</td>
<td>0.089</td>
<td>0.616</td>
<td>0.213</td>
</tr>
<tr>
<td>13</td>
<td>Absorption efficiency [%]</td>
<td>0.193</td>
<td>0.984</td>
<td>0.791</td>
</tr>
</tbody>
</table>
8.2.2 Training the ANN Models and Generating the EURECA Models

The ANN models were trained using the neural network algorithm provided by Rapidminer (Trademark of Rapidminer). The ANN models were trained using attribute values that were normalized to [-1, 1]. The activation functions for the hidden and output neurons used for the trained ANN models were the sigmoid and linear function, respectively. The learning rate and momentum used to train the ANN models were 0.3 and 0.2, respectively. Each of the trained ANN models had one hidden layer with four hidden neurons and was trained with 20-fold validation. The EURECA algorithm was implemented in the statistical language, R, and applied to extract rules from the trained ANN models. The application of EURECA requires input on the locations of the breakpoints, which were set to (-2.49, 2.49), and the rules or equations were extracted using the real values of the attributes. The accuracies of the trained ANN models and the EURECA models were measured by the mean square error (MSE) and $R^2$. The MSE measures the deviation of the predicted value from the actual value, and $R^2$ shows how close the actual values are to the fitted line. An accurate model will have an MSE value as low as possible and an $R^2$ value that is close to 1 as possible. The values of $R^2$ and MSE can be calculated as follows:

$$R^2 = 1 - \frac{\sum_i(y_i - \hat{y}_i)^2}{\sum_i(y_i - \bar{y})^2}$$

$$MSE = \frac{\sum_i(y_i - \hat{y}_i)^2}{n}$$

where $y_i =$ the expected output of the $i^{th}$ dataset, $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ and $\hat{y} =$ predicted output, $n =$ number of data tuples.
The accuracies of the trained ANN models and the generated EURECA models in terms of mean square error (MSE) and \( R^2 \), and the number of rules generated by each of the EURECA models are shown in Table 8.2.

As shown in Table 8.2, the generated EURECA models consistently demonstrated higher predictive accuracy than their underlying ANN models. The trained ANN models had \( R^2 \) values that ranged from 0.75 to 0.93 or 75% to 93%, whereas the generated EURECA models had \( R^2 \) values that ranged from 0.88 to 0.97 or 88% to 97%. All the MSE values given by the EURECA models were also lower than those given by the trained ANN models. The predictive accuracies of the EURECA models were compared to the results given by the models generated in (i) (Zhou et al., 2009) using the regression approach, (ii) (Wu & Chan, 2011) using the ANN combined with sensitivity analysis approach, and (iii) (Zhou et al., 2010) using the ANFIS approach. The results showed that the EURECA models demonstrated similar accuracies to the ANFIS approach in (Zhou et al., 2010). However, compared to the size of the rules sets generated, the EURECA algorithm gave significantly smaller rule sets than the ANFIS algorithm. Each of the ANFIS models had 144 rules, whereas the number of rules or equations given by the EURECA models ranged from 41 to 52. The comparison results described are summarized in Table 8.3.
Table 8.2 – Accuracies of the trained ANN and generated EURECA models

<table>
<thead>
<tr>
<th></th>
<th>MSE</th>
<th>$R^2$</th>
<th>No. of Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ANN</td>
<td>EURECA</td>
<td>ANN</td>
</tr>
<tr>
<td>$CO_2$ Production Rate</td>
<td>1.22E-03</td>
<td>5.58E-04</td>
<td>0.9366</td>
</tr>
<tr>
<td>Heat Duty</td>
<td>3.18E+07</td>
<td>2.39E+07</td>
<td>0.8400</td>
</tr>
<tr>
<td>Lean Loading</td>
<td>3.19E-03</td>
<td>8.89E-04</td>
<td>0.7593</td>
</tr>
<tr>
<td>Absorption Efficiency</td>
<td>2.14E-03</td>
<td>1.10E-03</td>
<td>0.9269</td>
</tr>
</tbody>
</table>

Table 8.3 – Comparing predictive accuracies and number of rules given by different modeling approaches

<table>
<thead>
<tr>
<th>Approach</th>
<th>Model Predictive Accuracy</th>
<th>No. of rules generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression (Zhou et al., 2009)</td>
<td>70 % - 80 %</td>
<td>N/A</td>
</tr>
<tr>
<td>ANN + SA (Wu &amp; Chan, 2011)</td>
<td>85 % - 99 %</td>
<td>N/A</td>
</tr>
<tr>
<td>ANFIS (Zhou et al., 2010)</td>
<td>88 % - 97 %</td>
<td>144</td>
</tr>
<tr>
<td>EURECA</td>
<td>88 % – 97 %</td>
<td>41 – 52</td>
</tr>
</tbody>
</table>
8.3 Solving the Operational Problems using the Generated EURECA Models

Since the generated EURECA models demonstrated satisfactory predictive accuracy and comprehensibility, they were applied to solving some of the operational problems encountered in the operation of the CO\(_2\) capture process system. Some example scenarios include: (i) when a flue gas with different CO\(_2\) concentration enters the absorber column, the operators would need to determine the value of amine circulation rate and steam flow rate through reboiler in order to maintain the level of CO\(_2\) production rate, (ii) when the CO\(_2\) absorption efficiency becomes abnormally low during operation, the operators would need to adjust the value of steam flow rate so that the absorption efficiency could reach the desired level again, while the values of the amine circulation rate and CO\(_2\) concentration flue gas remain unchanged. These sample scenarios involve setting some value for the some target predicted attribute values, such as CO\(_2\) production rate, and CO\(_2\) absorption efficiency, and the determination of some predictor attribute values, such as steam flow rate through reboiler, and amine circulation rate, so as to achieve the target predicted attribute values. We called this specific type of problems the “back calculation” problem, and we believed the EURECA algorithm is useful for solving this type of problems. Two approaches using the rules generated by the EURECA algorithm are proposed. The CO\(_2\) production dataset was used as an illustrative example in each of the proposed approaches. Both of which were implemented in the statistic language R. Both novice and expert approaches begin with the EURECA model of equations.
8.3.1 Novice Approach

The novice approach was designed for inexperienced operators or non-domain experts who are still learning about the operation process. It adopts a brute-force search or grid search approach, in which minimum inputs from the operators are involved. The operator only needs to specify the values of the known predictor attribute, the value of the predicted attribute that needs to be achieved and the number of increments in which he would like the values to be solved, then the values of the unknown predictor attributes will be generated. The details of the approach are explained with a sample scenario (labeled as sample scenario #1) shown in Table 4. In this example, the unknown predictor attributes were CO$_2$ concentration in the flue gas ($x_2$) and off gas flow rate ($x_3$).
Table 8.4 – Sample scenario #1 solved by the novice approach

<table>
<thead>
<tr>
<th>CO₂ production rate (Y)</th>
<th>x₁</th>
<th>x₂</th>
<th>x₃</th>
<th>x₄</th>
<th>x₅</th>
<th>x₆</th>
<th>x₇</th>
<th>x₈</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.7268</td>
<td>0.9519</td>
<td>?</td>
<td>?</td>
<td>10.99781</td>
<td>19.81884</td>
<td>499.4374</td>
<td>63.11688</td>
<td>7.0192</td>
</tr>
</tbody>
</table>
The steps involved in solving for these values using the novice approach are as follows:

1. Identify the matching rules or equation

Since each equation covers a specific range of values for the predictor and predicted attributes, as shown in Eq 8.1 below, for a given operational scenario only a limited number of equations will cover all the parameter values that are specified by the operator.

Rule #1: IF

\[
0.8515 \leq \text{Mass flow of CO}_2 \text{ in the flue gas (x1)} \leq 0.9712 \\
10.58 \leq \text{CO}_2 \text{ concentration in the flue gas (x2)} \leq 11.66 \\
3.344 \leq \text{Off gas flow rate (x3)} \leq 4.179 \\
7.888 \leq \text{Amine circulation rate (x4)} \leq 12.02 \\
17.80 \leq \text{Reboiler pressure (x5)} \leq 42.94 \\
292.9 \leq \text{Pressure of steam through reboiler (x6)} \leq 502.4 \\
66.27 \leq \text{Steam flow rate through reboiler (x7)} \leq 79.34 \\
4.050 \leq \text{Amine concentration (x8)} \leq 6.996
\]

Then

\[
\text{CO}_2 \text{ production rate (y) = 0.1864}_1 + 0.0099x2 + 0.0052x3 + 0.0012x4 + 0.0023x5 + \\
0.0003x6 + 0.0005x7 - 0.0030x8 + 0.2617
\]

Eq. (8.1)

These matching equations will be used to solve for the values of the unknown parameters. For sample scenario #1, rule or equation #14 and #23 were found as the matching equations as shown in Figure 8.2.

2. For each of the matching equations identified, perform the following:
a. Identify the range of values of the unknown attributes covered by the equations. In sample scenario #1, the range of values covered by rule #14 was [10.3510, 11.2847] and [3.5179, 4.1526] for the CO$_2$ concentration in the flue gas ($x_2$) and off gas flow rate ($x_3$), respectively, as shown in Figure 8.3.

b. List the values between the identified ranges in the number of increments specified by the operator. For example, if the specified number of increment is 10, then the lists will consist of 10 equally distributed values between the ranges identified. The list of values for CO$_2$ concentration in the flue gas ($x_2$) was (10.3510, 10.4547, 10.5585, 10.6622, 10.7660, 10.8697, 10.9735, 11.0772, 11.2847) and the list of values for off gas flow rate ($x_3$) was (3.5179, 3.5884, 3.6589, 3.7295, 3.8000, 3.8705, 3.9410, 4.0116, 4.0821, 4.1526).

c. Test all value combinations given by the lists and calculate the predicted value using the matching equation. In the sample scenario #1 of two unknowns ($u$) and 10 increments ($i$), a total of $10^2$ or 100 combinations will be tested. In other words, for each equation, the number of combination is $i^u$.

d. The combination that returns a calculated predictor value closest to the desired target predictor value is selected as the solution.

3. Repeat step (2) for all matching equations.

4. When there are more than one matching equations, the solutions are ranked according to the difference between the calculated and actual predictor value, from lowest to highest, as shown in Figure 8.4, and the operator can select the appropriate solution accordingly.
**Figure 8.2** – Matching equations found for sample scenario #1

<table>
<thead>
<tr>
<th>Rules</th>
<th>No of data</th>
<th>Intercept</th>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
<th>X5</th>
<th>X6</th>
<th>X7</th>
<th>X8</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.0000</td>
<td>896.0000</td>
<td>0.6506</td>
<td>0.0237</td>
<td>-0.0005</td>
<td>-0.0831</td>
<td>0.0010</td>
<td>0.0270</td>
<td>0.0001</td>
<td>-0.0017</td>
<td>-0.0172</td>
</tr>
<tr>
<td>23.0000</td>
<td>169.0000</td>
<td>2.9098</td>
<td>-1.6294</td>
<td>0.1731</td>
<td>-0.8267</td>
<td>0.0226</td>
<td>-0.0026</td>
<td>0.0005</td>
<td>0.0124</td>
<td>-0.1053</td>
</tr>
</tbody>
</table>

**Figure 8.3** – Range of values covered by Rule #14 for sample scenario #1

| Rules | No of data | Y1Min | Y1Max | X1Min | X1Max | X2Min | X2Max | X3Min | X3Max | X4Min |
|-------|------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| 14.0000 | 896.0000 | 0.4533 | 0.8936 | 0.6048 | 0.9888 | 10.3510 | 11.2847 | 3.5179 | 4.1526 | 7.9501 |
| 23.0000 | 169.0000 | 0.3483 | 0.9967 | 0.7821 | 0.5936 | 10.8271 | 11.4764 | 3.6617 | 4.1401 | 7.9754 |

**Figure 8.4** – Solutions given by the novice approach for sample scenario #1

<table>
<thead>
<tr>
<th>Matching Equation</th>
<th>X2</th>
<th>X3</th>
<th>Calculated Output</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.0000</td>
<td>10.3510</td>
<td>3.8000</td>
<td>0.7264</td>
<td>0.0005</td>
</tr>
<tr>
<td>23.0000</td>
<td>11.1149</td>
<td>3.6617</td>
<td>0.7246</td>
<td>0.0023</td>
</tr>
</tbody>
</table>
8.3.1.1 Performance Evaluation of the Novice Approach

To test the accuracy of the novice approach, the scenarios of 2, 3, 4 and 5 unknowns with 10, 20 and 30 number of increments were tested. A total of 11 test scenarios were generated because R crashed with an out-of-memory fatal error when testing the scenario of 5 unknowns and 30 numbers of increments. Thirty test cases were randomly selected from the CO₂ production rate dataset to test each of the scenarios, and the unknown attributes were also randomly selected in each test scenario.

To evaluate the performance of the novice approach, the following indicators were used:

1. The average absolute deviation of the calculated predicted attribute values, \( E_{y0} \)
2. The average absolute deviation of the calculated predictor attribute values, \( E_{x0} \)

Both the values of \( E_{y0} \) and \( E_{x0} \) evaluate the difference between the calculated given by the novice approach and target values, and they reflect how accurate the given solutions are. The average absolute deviation (AAD) is given by percentage difference between the target and calculated value and can be expressed as the following equation:

\[
AAD (\%) = \text{Absolute} \left( \frac{\text{Target Value} - \text{Calculated Value}}{\text{Target Value}} \right) \times 100\%
\]

The average absolute deviation between the calculated and target predicted attribute values, \( E_{y0} \), is shown in Table 8.5 and the plot of \( E_{y0} \) verse number of unknown variables is shown in Figure 8.5. The desired \( E_{y0} \) is as close to zero as possible to indicate the calculated value is close to the target value of the predicted attribute.
Table 8.5 – The AAD between the calculated and target predicted attribute given by the test cases

<table>
<thead>
<tr>
<th>Number of Unknown</th>
<th>Number of Increment</th>
<th>Average $E_{y0}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>0.6315</td>
<td>0.0122</td>
</tr>
<tr>
<td>3</td>
<td>0.0214</td>
<td>0.0011</td>
</tr>
<tr>
<td>4</td>
<td>0.0004</td>
<td>0.0001</td>
</tr>
<tr>
<td>5</td>
<td>0.0001</td>
<td>0.0000</td>
</tr>
<tr>
<td>Average $E_{y0}$</td>
<td>0.1633</td>
<td>0.0034</td>
</tr>
</tbody>
</table>
Figure 8.5 – ADD between the calculated and target predicted attribute Vs number of
unknown attributes
When there are more than one matching equations in a particular problem scenario, more than one solutions will be given, and with no knowledge on which equation was the correct equation, i.e. the equation that the given test case actually belongs to, the equation and combination that gave the lowest AAD error between the calculated and target predicted attribute was chosen to be the solution. As shown in Figure 8.5, the average absolute error of the calculated predicted attribute decreased as the number of unknown variables increased. The AAD between the calculated and target predicted attribute when there were 2 unknown attributes was 0.2535%, and the error rate was decreased to 0.0000% AAD when there were 5 unknown attributes. The low values of $E_{y0}$ indicated high accuracy of the calculated predictor attribute. The number of increments, however, did not seem to have a direct linear relationship with the value of $E_{y0}$. As shown in Table 8.5, the average $E_{y0}$ at 10 increments was 0.1633%, the values decreased to 0.0034% at 20 increments, and the value increased again at 30 increments to 0.0968%.

The average absolute deviation between the calculated and target predictor attribute values, $E_{x0}$, is shown in Table 8.6 and the plot of $E_{x0}$ verse number of unknown variable is shown in Figure 8.6. The desired $E_{x0}$ is as close to zero as possible to indicate minimum deviation between the calculated and target value of the predictor attribute.
Table 8.6 – The AAD between the calculated and target predicted attribute given by the test cases

<table>
<thead>
<tr>
<th>Number of Unknown</th>
<th>Number of Increment</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>Average $E_{x0}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>10.5832</td>
<td>14.4816</td>
<td>9.0850</td>
<td>11.38</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>14.3928</td>
<td>15.6213</td>
<td>13.4164</td>
<td>14.48</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>13.7583</td>
<td>17.2613</td>
<td>--</td>
<td>15.51</td>
</tr>
<tr>
<td>Average $E_{x0}$</td>
<td></td>
<td>12.0431</td>
<td>15.0660</td>
<td>11.1911</td>
<td></td>
</tr>
</tbody>
</table>
Figure 8.6 – ADD between the solved and actual predicted attribute Vs number of unknown attributes
Although Figure 8.6 showed that the relationship between AAD and the number of unknown attributes was not directly linear, the plot showed a trend of increasing $E_{x0}$ as the number of unknown predictor attribute increased. As shown in Table 6, when there were 2 unknown, the AAD between the solved and actual predictor attribute values was 11.38%, whereas when there were 5 unknown predictor attributes, the value of AAD was increased to 15.51%. However, unlike $E_{y0}$, the highest $E_{x0}$ was given when the number of increments was 20. As shown in Table 6, the average $E_{x0}$ at 10 increments was 12.04%, the value increased to 15.07% when increment was increased to 20 and the value decreased to 11.19% at 30 increments. Therefore, similar to $E_{y0}$, the number of increments did not seem to have a direct linear relationship with the value of $E_{x0}$.

The results showed that even with the minimum information given by the operator the novice approach could generate reliable results. This approach is useful to assist inexperienced operators to monitor the process plant and the generated rules could also provide the operator a better understanding on the relationship between the different operating parameters.

### 8.3.2 Expert Approach

Unlike the novice approach, the expert approach requires more input from the operator or domain expert. This approach would be suitable for experienced operators and domain experts who would prefer having control over the problem-solving process. Therefore, two benefits of this approach are: (i) better control of the problem-solving process, (ii) user defined granularity of the error is supported.
The tasks that require the expert’s input are listed below:

1. Selecting the equation for solving the unknown values.
2. Estimating some of the unknown attribute values in order to solve all the values.
3. Specifying the allowable error on the target predictor value.
4. Specifying the increment in the allowable error in which the unknown values will be solved.

The allowable error on the target predicted value and the value of increment refer to the amount of error the calculated predicted attribute can have and the values that will be used to solve for the unknown values. For example, assuming the target predicted attribute value is Y and the expert specifies an allowable error of 5% and an increment of 1%, then the unknown values will be solved based on the value of Y +/- 5% at every 1%, i.e. Y +/- 1%, Y +/- 2%, Y +/- 3%, Y +/- 4% and Y +/- 5%, as shown in Figure 8.7.

The first two tasks of selecting the equation and estimating the unknown values rely heavily on the expert’s domain knowledge. Some heuristics that can assist the expert to in selecting the equation include: (1) if a particular equation covers a reasonable range of values for the unknown attribute, and (2) if a particular equation provides a solution within a reasonable allowable error. For both of these heuristics, the “reasonable range” is subjectively defined by the expert. Based on the ranges of values covered by the selected equation, the expert could make an estimate on some of the unknown attribute based on his knowledge of the given scenario. The current version of the expert approach supports solving up to two unknowns values.
Figure 8.7 – Allowable error of 5% and an increment of 1%
A detailed example (labeled as sample scenario #2) is given below to illustrate the expert approach in solving the back calculation problem using a data tuple from the CO$_2$ production rate dataset. The randomly selected data tuple belongs to Rule #1 and the two unknown predictor variables were steam flow rate through reboiler ($x_7$) and amine circulation ($x_8$). The target values of $x_7$ and $x_8$ were 73.8965 and 5.1800, respectively. The attribute values of the selected data tuple are shown in Table 8.7.
Table 8.7 – Sample scenario #2 solved by the expert approach

<table>
<thead>
<tr>
<th>CO₂ production rate (Y)</th>
<th>x₁</th>
<th>x₂</th>
<th>x₃</th>
<th>x₄</th>
<th>x₅</th>
<th>x₆</th>
<th>x₇</th>
<th>x₈</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.7889</td>
<td>0.9096</td>
<td>11.1140</td>
<td>3.7477</td>
<td>8.0032</td>
<td>29.0970</td>
<td>336.9597</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Twelve matching equations that matched the values of all the attributes were found, as shown in Figure 8.8. With a user-defined allowable error of 5% and an increment of 1%, only 4 out of the 12 equations provided a solution and Rule #1 was among one of the equations that provided a solution. The criterion of whether an equation provides a solution within a user-defined error rate reduced the number of selected equations by 60%. The values covered by the remaining four equations for the two unknown attributes are shown in Table 8.8. Based on the range of values and the expert’s domain knowledge, he could select an equation to solve for the two unknowns.
**Table 8.8 – Range of values covered by the matching equations with solutions for sample scenario #2**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Steam flow rate through reboiler ($x_7$)</th>
<th>Amine circulation ($x_8$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>66.273 – 79.345</td>
<td>4.050 – 6.996</td>
</tr>
<tr>
<td>#2</td>
<td>66.925 – 79.905</td>
<td>4.933 – 8.009</td>
</tr>
<tr>
<td>#3</td>
<td>64.42 – 77.572</td>
<td>4.820 – 5.329</td>
</tr>
<tr>
<td>#12</td>
<td>68.057 – 79.228</td>
<td>4.990 – 7.187</td>
</tr>
</tbody>
</table>
Assuming the domain expert selected Rule #1, which was the rule that the test data tuple belonged to, he then would have to estimate one of the unknown values in order to solve for the other one. Assuming the expert estimated that $x_7 = 70$, then at an allowable error of $Y +/- 5\%$ and an increment of $1\%$, one solution of $x_8 = 5.78$ was given at $Y - 3\%$. As shown in Figure 8.9 the solution of $x_8 = 5.78$ was the only solution that lied within the region bounded by the minimum and maximum values of $x_7$ and $x_8$ covered by Rule #1.

A solution was only considered as valid if it lied within the bounded region. The line labeled as “$Y - 2\%$” in Figure 8.9 was the equation given at $Y - 2\%$ and the solved value of $x_8$ (indicated by circle on this line) was outside of the bounded region. Therefore the solved value given at $Y - 2\%$ was considered as an invalid solution. The line labeled as “$Y - 3\%$” was the equation given at $Y - 3\%$ and the solved value of $x_8$ lied within the bounded region and therefore was considered as a valid solution.

Based an estimated value of $x_7 = 70$, the average absolute deviation (AAD) of the solved valued of $x_8$ was:

$$E_{x8} = \left| \frac{5.18 - 5.78}{5.18} \right| \times 100\% = 11.58\%$$

and it was given at $Y - 3\%$ which means the average absolute deviation (ADD) between the calculated and actual predicted attribute, $E_{y0} = 3\%$. 
Figure 8.9 – Solutions given by Rule #1 with allowable error of $Y \pm 5\%$ and increment of $1\%$
The expert is given the flexible control over definition of the allowable error in the output or predicted attribute value. An alternative scenario can illustrate this flexibility when the expert wished to refined the solution by decreasing the size of increment. Based on sample scenario #2, **Figure 8.10** shows the solutions given when the increment was decreased to 0.1% using the similar calculation procedure as discussed when the increment was 1%. Since the increment was more refined, more valid solutions were found within the bounded region and these solutions were given between the error rate of -2.3 % to -3.3 %. The circles within the bounded region shown in Figure 8.10 indicate the valid solutions for $x_8$. 
Figure 8.10 – Solutions given by Rule #1 with allowable error of Y +/- 5% and increment of 0.1% for sample scenario #2
8.3.2.1 Performance Evaluation of the Expert Approach

Twenty-nine more test cases were randomly selected from the CO₂ production rate dataset to test the expert approach, and two predictor attributes were randomly selected in each case as the unknown attributes. The two predictor attributes were labeled as “Unknown 1” and “Unknown 2”. “Unknown 1” were assumed to be the expert estimated attribute, however the values of unknown 1 were taken from the real dataset. Hence only “Unknown 2” were the system calculated predictor attribute values. The indices used to evaluate the performance of the proposed approach include:

1. The number of matching equations identified in each test scenario, $T_{\text{eqn}}$

2. The number of matching equations that provided a solution with the allowable error of $Y+/-5\%$ and an increment of $1\%$, $T_{\text{eqn}, 5\%}$

Both the value of $T_{\text{eqn}}$ and $T_{\text{eqn}, 5\%}$ measures the ease of the matching equation selection process for the domain expert. The desired values of $T_{\text{eqn}}$ and $T_{\text{eqn}, 5\%}$ are as close to 1 as possible.

3. The average absolute deviation (ADD) of the calculated predicted attribute values, $E_{y0}$

4. The average absolute deviation (ADD) of the solved predictor attribute values, $E_{x0}$

Both the values of $E_{y0}$ and $E_{x0}$ evaluate the difference between the calculated given by the approach and target values, and they reflect how accurate the given solutions are. The desired values of $E_{y0}$ and $E_{x0}$ are as close to 0\% as possible, indicating minimum deviation between the calculated and target value.
Following the procedure discussed when the error rate was at 5% with a 1% increment, the results generated are shown in Table 8.9. The result of sample scenario #2 was included as case #1.

As shown in Table 8.9, on average 9 matching equations were found in the test scenarios. By examining whether a solution was given at the allowable error rate of $Y +/- 5\%$, the number of matching equations were reduced to 5 on average. Based on an accurate estimation of unknown 1, the average absolute deviation of the solved value, $E_x$, was 14.12% and the average absolute deviation between the calculated and target predicted attribute, $E_y$, was 2.53%. In case #13, no solution was found when the allowable error rate was at $Y +/- 5\%$. A solution was given, however, when the error rate was increased to 6% for this test case. The results showed that the expert approach is capable of providing a reliable solution given the expert’s correct judgments on the equation used and estimated value of one unknown predictor attribute.
Table 8.9 – Results of solving the CO$_2$ production rate back calculation problems using the expert approach

<table>
<thead>
<tr>
<th>Case #</th>
<th>Unknown 1</th>
<th>Unknown 2</th>
<th>$T_{eqns}$</th>
<th>$T_{eqns, 5%}$</th>
<th>$E_{x0}$ (%)</th>
<th>$E_{y0}$ (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$x_7$</td>
<td>$x_8$</td>
<td>12</td>
<td>4</td>
<td>11.58</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>$x_8$</td>
<td>$x_5$</td>
<td>10</td>
<td>7</td>
<td>28.5</td>
<td>2.4</td>
</tr>
<tr>
<td>3</td>
<td>$x_4$</td>
<td>$x_3$</td>
<td>11</td>
<td>5</td>
<td>10.22</td>
<td>4.1</td>
</tr>
<tr>
<td>4</td>
<td>$x_2$</td>
<td>$x_4$</td>
<td>9</td>
<td>7</td>
<td>0.07</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>$x_3$</td>
<td>$x_5$</td>
<td>11</td>
<td>4</td>
<td>39.47</td>
<td>3.04</td>
</tr>
<tr>
<td>6</td>
<td>$x_4$</td>
<td>$x_5$</td>
<td>12</td>
<td>9</td>
<td>23.23</td>
<td>5.8</td>
</tr>
<tr>
<td>7</td>
<td>$x_1$</td>
<td>$x_7$</td>
<td>8</td>
<td>6</td>
<td>5.21</td>
<td>2.2</td>
</tr>
<tr>
<td>8</td>
<td>$x_4$</td>
<td>$x_7$</td>
<td>12</td>
<td>6</td>
<td>4.16</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>$x_2$</td>
<td>$x_3$</td>
<td>11</td>
<td>5</td>
<td>19.9</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>$x_1$</td>
<td>$x_2$</td>
<td>9</td>
<td>7</td>
<td>1.09</td>
<td>0.45</td>
</tr>
<tr>
<td>11</td>
<td>$x_4$</td>
<td>$x_3$</td>
<td>10</td>
<td>4</td>
<td>4.24</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>$x_8$</td>
<td>$x_5$</td>
<td>10</td>
<td>10</td>
<td>16.67</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>$x_2$</td>
<td>$x_7$</td>
<td>9</td>
<td>0</td>
<td>No Sol</td>
<td>No Sol</td>
</tr>
<tr>
<td>14</td>
<td>$x_4$</td>
<td>$x_7$</td>
<td>12</td>
<td>4</td>
<td>8.57</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>$x_5$</td>
<td>$x_4$</td>
<td>5</td>
<td>4</td>
<td>9.26</td>
<td>0.6</td>
</tr>
<tr>
<td>16</td>
<td>$x_8$</td>
<td>$x_3$</td>
<td>12</td>
<td>4</td>
<td>13.6</td>
<td>2</td>
</tr>
<tr>
<td>17</td>
<td>$x_1$</td>
<td>$x_5$</td>
<td>5</td>
<td>2</td>
<td>4.39</td>
<td>2</td>
</tr>
<tr>
<td>18</td>
<td>$x_1$</td>
<td>$x_7$</td>
<td>5</td>
<td>3</td>
<td>21.15</td>
<td>4</td>
</tr>
<tr>
<td>19</td>
<td>$x_2$</td>
<td>$x_3$</td>
<td>2</td>
<td>4</td>
<td>11.74</td>
<td>4</td>
</tr>
<tr>
<td>20</td>
<td>$x_6$</td>
<td>$x_1$</td>
<td>5</td>
<td>2</td>
<td>8.1</td>
<td>1.35</td>
</tr>
<tr>
<td>21</td>
<td>$x_5$</td>
<td>$x_1$</td>
<td>12</td>
<td>1</td>
<td>3.45</td>
<td>3</td>
</tr>
<tr>
<td>22</td>
<td>$x_7$</td>
<td>$x_6$</td>
<td>5</td>
<td>2</td>
<td>0.12</td>
<td>3</td>
</tr>
<tr>
<td>23</td>
<td>$x_6$</td>
<td>$x_8$</td>
<td>8</td>
<td>5</td>
<td>68.97</td>
<td>4</td>
</tr>
<tr>
<td>24</td>
<td>$x_5$</td>
<td>$x_1$</td>
<td>8</td>
<td>6</td>
<td>13.45</td>
<td>0.2</td>
</tr>
<tr>
<td>25</td>
<td>$x_5$</td>
<td>$x_1$</td>
<td>10</td>
<td>10</td>
<td>24.71</td>
<td>1.65</td>
</tr>
<tr>
<td>26</td>
<td>$x_2$</td>
<td>$x_1$</td>
<td>10</td>
<td>5</td>
<td>29.36</td>
<td>1.45</td>
</tr>
<tr>
<td>27</td>
<td>$x_4$</td>
<td>$x_7$</td>
<td>14</td>
<td>3</td>
<td>1.41</td>
<td>1</td>
</tr>
<tr>
<td>28</td>
<td>$x_5$</td>
<td>$x_8$</td>
<td>14</td>
<td>7</td>
<td>15.09</td>
<td>2.5</td>
</tr>
<tr>
<td>29</td>
<td>$x_3$</td>
<td>$x_7$</td>
<td>3</td>
<td>5</td>
<td>3.13</td>
<td>3</td>
</tr>
<tr>
<td>30</td>
<td>$x_7$</td>
<td>$x_2$</td>
<td>8</td>
<td>5</td>
<td>8.71</td>
<td>2.2</td>
</tr>
</tbody>
</table>

**Average** | **9.07** | **5.00** | **14.12** | **2.53**
8.4 Discussion

8.4.1 Comparison between the Novice and Expert Approach

In the novice approach, all the steps involved were automated and minimum input was required from the process operator or domain expert. On the other hand, the expert approach allowed the domain expert to have control over the problem-solving process. However the accuracy of the solution relied heavily on the domain expert’s knowledge.

The main differences between the two approaches include:

1. On the selection of rules:

   The novice approach attempts to solve the problems using all matching rules or equations whereas the expert approach solves the problem using the expert-selected equation among the identified matching equations. The domain expert could examine the equations and determine whether an equation covers a reasonable range of values or provides a solution within a reasonable error rate. In the case where two unknown predictor attributes were involved, the novice approach found an average of 4.39 matching equations with solutions and the expert approach found an average of 9 matching equations with 5 of them providing a solution within 5% error.

2. On the number of unknown predictor attributes:

   As illustrated by the application to the CO$_2$ production rate dataset, the novice approach was able to solve up to 5 unknown predictor attribute values with up to 20 increments, whereas the current version of the expert approach was only able to solve problems with two unknown predictor attributes.
3. On the problem-solving process:

The novice approach solves the values using brute-force search: first, it lists all the values by the user-specified increment between the value ranges covered by the matching equation; then, it tests all the combinations of the listed values. In comparison, the expert approach requires the expert to estimate one of the unknown values based on the value range covered by the equation he selected, and then the remaining unknown value can be solved. In the two unknown cases, the solutions given by the novice approach had an average absolute deviation (AAD) of 0.2535% between the calculated and target predicted attribute values and an AAD of 11.39% between the solved and actual predictor attribute values. In comparison, the error given by the expert approach was higher with an AAD of 2.53% between the calculated and target predicted attribute values and 14.12% between the solved and actual predictor attribute values.

4. On control over error refinement:

The novice approach offers the user no control over the error refinement during the problem-solving process; it only ranks the solutions based on the deviation between the calculated and target predicted attribute, and the solution with the lowest deviation is assumed to be the best solution. On the other hand, the expert approach provides the user flexible control over the allowable error between the calculated and target attribute, as well as the increment on the error rate. As demonstrated in sample scenario #2, solutions with lower error could be found when the increment of error was decreased from 1% to 0.1%.
8.4.2 Comparison between the EURECA and ANFIS approach

An ANFIS approach was proposed in (Zhou et al., 2013) in order to solve four operational problems in the CO₂ capture process system at CETRI. Each case scenario had a different set of predictor and predicted attributes, and one fuzzy inference system was trained for each of the operational problems. A summary of the input (I) and output (O) attributes used for generating the fuzzy inference systems is shown in Figure 8.11. When the attribute of flow rate of steam through reboiler was used as the output as showed in Figure 11, the ANFIS algorithm failed to generate a model for both scenario #1 and scenario #2 (Zhou et al., 2013).

However, by removing the attribute flow rate of steam through reboiler in both scenario #1 and #2 as output attribute, and replaced it as an input attribute, the fuzzy inference models were successfully generated. Based on these results, Zhou et al. (2013) concluded that the attribute steam flow rate through reboiler was a critical input parameter that could not be omitted. In other words, the ANFIS approach was not able to solve operational problems in which the value of steam flow rate through reboiler needed to be determined.
Figure 8.11 – Input and output attributes for the four operational scenarios used in

(Zhou et al., 2013)
By contrast, the EURECA approach has the following advantages. First, the EURECA approach does not require new models to be trained for different operational scenarios. When the EURECA models have been generated for a set of predictor and predicted attributes, they can be used to solve any operational problems that involve the same set of attributes. As demonstrated in this study, the same EURECA model generated for the CO₂ production rate was used to solve all the operational problems that involved the CO₂ production rate. With the novice approach, the model could be used to determine up to 5 unknown values of the predictor attributes. Secondly, since no new model is required for different operational problems, the problem of re-selecting appropriate input and output attributes was avoided. The ANFIS approach failed to fully resolve the operational problems when the steam flow rate through reboiler was set as an output attribute. But for the EURECA approach, this attribute can be the unknown predictor in the back calculation problem and can be solved. As shown in Table 8.9, some of the expert approach test cases involved the steam flow rate of reboiler \( x_7 \), and all the test scenarios of the novice approach involved cases with the steam flow rate of reboiler as unknown attribute.

### 8.4.3 Drawbacks of the EURECA Approach

Despite the advantages and reliable performances of the EURECA approaches, there are some drawbacks and areas that needed to be improved. These include:

1. The number of unknown attributes that the novice approach could solve in this study was limited by the computing resources. Although this problem could be easily resolved by upgrading the computing resources, this shows that the proposed
brute-force search approach can be time and resource consuming and may not be the most efficient approach.

2. The number of unknown attributes that the expert approach could solve needs to be improved. The current version could only solve two unknowns but in reality, the operational problems usually involve more than two. The expert approach can be extended so that the domain expert can specify the unknown values that he wishes to estimate, and the remaining unknown could be solved using the brute-force search in the novice approach.

3. Some of the operational problems involve more than one of the four predicted attributes, however, the proposed approaches in this study could only solve problems involving the same predicted attributes. To improve this, either the ANN model needs to be retrained so that more than one of the predicted attributes are involved or the EURECA models need to be combined to solve problems involving more than one predicted attributes.

8.5 Conclusion and Future Work

With sixteen components and over 140 parameters involved, the operation of the CO2 capture system is a complicated task. During operations, the operators often need to determine the values of some process parameters in order to achieve the desired system performances. To solve this back calculation problem, this study proposed using the rules extracted by the EURECA algorithm. The EURECA algorithm is an ANN rule extraction algorithm that generates rules in the form of multiple linear regression equations from trained ANN model by clustering the dataset using the weighted input values. Four ANN
models were trained, one for each of the four predicted attributes: (1) CO₂ production rate, (2) heat duty, (3) lean loading and (4) CO₂ absorption efficiency using the ANN algorithm in Rapidminer. The predictor attributes used include: (1) mass flow of CO₂ in the flue gas, (2) CO₂ concentration in the flue gas, (3) off gas flow rate, (4) amine circulation rate, (5) reboiler pressure, (6) pressure of steam through reboiler, (7) Steam flow rate through reboiler, and (8) amine concentration. The trained ANN models had a predictive accuracy of 75% – 94%. The EURECA algorithm was then applied to the trained ANN models with the breakpoints locating at (-2.49, 2.49) and the generated EURECA models for the CO₂ capture process systems gave a predictive accuracy between 88% – 97%.

To solve the operational problem two approaches were proposed, and the CO₂ production rate EURECA model as an illustration. The first approach proposed was the novice approach which solves the unknown values of the predictor attributes with minimum input from the operator. The approach solves for the unknown predictor attributes via a brute-force search approach and the solution that give the lowest average absolute deviation between the calculated and target predicted attribute was selected as the solution. The second approach proposed was the expert approach, in which the operator or domain expert has control over the problem-solving process through selection of equations used, estimation of the value of some of the unknowns, and the desired error rate. The results showed that the novice approach was capable of solving up to five unknown predictor attributes values with 20 increments, whereas the expert approach was able to solve two unknown values only. The solutions given by the novice approach had
an average absolute deviation (AAD) of 0.2535% between the calculated and target predicted attribute values and an AAD of 11.39% between the solved and actual predictor attribute values. In comparison, the solutions given by the expert approach was slightly less accurate with an AAD of 2.53% between the calculated and target predicted attribute values and 14.12% between the solved and actual predictor attribute values. Although the novice approach demonstrated higher accuracy, the expert approach has the major advantages of supporting (i) better control on the problem-solving process, and (ii) user defined granularity of error. Comparing the EURECA approach to previously proposed neuro-fuzzy (ANFIS) approach in (Zhou et al., 2013), the EURECA approach could use the same model to solve different operational problems involving the same set of attribute, whereas the ANFIS approach requires a new model to be trained for different problems. In addition, the selection of predictor parameters was critical when training the new ANFIS models, and if they were not correctly selected, the ANFIS approach would not be able to generate a model. This problem was completely avoided with the EURECA approach.

Some directions of future works include:

1. Extend the number of unknowns predictor attributes that can be solved by the expert approach,

2. Combine EURECA models to solve operational problems that involve more than one of the four key parameters of \( \text{CO}_2 \) production rate, heat duty, lean loading, and \( \text{CO}_2 \) absorption efficiency.
CHAPTER 9 CASE STUDY #3 UNDERSTANDING AND MONITORING AMINE SOLVENT IN THE CO₂ CAPTURE SYSTEM

To ensure the efficiency of an amine-based carbon dioxide capture plant, the concentration of the amine solvent is an important parameter. The state of the amine solvent is directly reflected by its physical and chemical properties such as density, viscosity and refractive index. However, the process of determining the chemical and physical properties of the amine solvent can be expensive and time-consuming, moreover if the solvent sample was not handled properly, the results can be inaccurate. Therefore, the application of machine learning algorithms, specifically artificial neural networks (ANN), was proposed by researchers as a way to provide an online and cost-effective solution for determining the properties of the amine solvent and monitoring the concentrations of the components. In this chapter, we proposed applying an ANN rule extraction algorithm called EURECA, as an alternative approach to traditional ANN algorithms, for predicting the physical properties of the amine solvent. The EURECA algorithm was able to not only provide highly accurate predictive capability similar to traditional feedforward backpropagation neural networks, it also had the advantage of generating explicit relationships between the predictor and predicted variables, which can enhance the domain expert’s understanding of the studied problem. The extracted rules can also be used to solve an often encountered operational problem called back calculation. The back calculation scenario describes the situation in which the user wishes to achieve a desired predicted variable value but lacks sufficient information on the values of all the predictor variables. The problem then involves determining the values of the unknown predictor parameters in such a way that the target predicted value
can be achieved. In the case of amine solvent monitoring, often the target predicted variable is the measured chemical and physical properties, such as density, viscosity, heat capacity and thermal conductivity, and the known predictor variable is temperature and the goal is to determine the unknown values of the predictor variables related to the composition of the solvent. Hence, the question becomes one of determining the composition of the solvent that will result in the value of the chemical and physical properties at a given temperature. The dataset used in this study was the experimental data obtained from (Pouryousefi et al. 2016; Pouryousefi, 2014). The solvent mixture was made up of monoethanolamine (MEA), carbon dioxide (CO₂), methyldiethanolamine (MDEA) and water. The chemical and physical properties measured at different temperature that ranged between 298 K to 323 K include: density, viscosity, refractive index, heat capacity, thermal conductivity, and thermal diffusivity.

By applying the EURECA algorithm, this study aims to achieve the following:

1. To provide better understanding on the relationships between the compositions of the amine solvent and its chemical and physical properties, including density, viscosity, refractive index, heat capacity, thermal conductivity, and thermal diffusivity.

2. To assist in monitoring of the amine solvent concentration in the carbon dioxide capture process system by solving the back calculation problem.
9.1 Background

9.1.1 Empirical Approach on Modeling Amine Solvents’ Properties

Various studies have been conducted in order to formulate correlation models for predicting the physical properties of the amine solvent mixture. The most extensively used empirical correlations defined for different physical properties include: Weiland correlation for density, Nissan-Grunburg correlation for viscosity, Gladston-Dale correlation for refractive index, and Redlich-Kister correlation for heat capacity, refractive index, and thermal conductivity (Pouryousefi et al., 2014). Amundsen et al. (2009) developed a correlation model for density of MEA + water + CO\textsubscript{2} measured between 25\textdegree C and 80 \textdegree C based on the Weiland correlation; the modeling results had a maximum error of 1.6\% relative deviation. Fu et al. (2012) suggested a modified Grunberg-Nissan equation for predicting the viscosity of carbonated MDEA-MEA aqueous solutions. They measured the viscosity of the carbonated mixed aqueous solution at 293.15 – 343.15 K, 0.2 – 0.5 amine mole fraction, and 0 – 0.5 mol CO\textsubscript{2}/mole amine. Pouryousefi et al. (2008) measured the refractive indices of three binary systems of MEA + water, MDEA + water and MEA + MDEA, and two ternary systems: MEA + water +CO\textsubscript{2} and MDEA + water + CO\textsubscript{2}, at different temperatures through experiments. Then they applied the correlation equations of: Gladston-Dale, Lorentz-Lorenz, Weiner, Heller and Arago-Biot. Their results showed that the Gladston-Dale correlation gave the best fit for both the binary and ternary systems, with an average absolute deviation of less than 0.1\% and 1.6\% for the two systems, respectively. Mehra (2003) measured the refractive indices of some binaries hexadecane and heptadecane with n-alkanols and their application of the Gladston-Dale, Heller, Lorentz-Lorenz, Argo-Biot, and Weiner
equations to the collected experimental data all gave good modeling results. Mundhwa et al (2007) applied the Redlich-Kister correlation equation to model the excess molar heat capacities of aqueous alkanolamines, which included 3-amino-1propanol, 2-(methylamino)-ethanol (MEA), and 1-amino-2-propanal, and their results showed that the Redlich-Kister equation was capable of modeling such systems with less than 0.23% of absolute percentage deviation. Chen et al. (2001) also used the Redlich-Kister correlation in their study to model the excess molar heat capacities of aqueous mixture. Their results showed that the Redlich-Kister equation was able to model the binary and ternary systems with an average absolute percentage deviation of 3.6% and 13.2%, respectively.

9.1.2 Artificial Neural Network Approach

Artificial neural network (ANN) is a machine learning algorithm that is capable of modeling non-linear and complex problems with high predictive accuracy. Various studies were conducted using the ANN algorithm to predict various properties of the amine solvent given the solvent’s composition. Mehlman et al. (1998) predicted density, viscosity, and refractive index of different ternary and quaternary solvent systems using ANN. The results showed less than 1 % of relative standard error for density and refractive index and less than 15 % for viscosity. The ANN models developed by (Kubendran, 2008) could predict density, viscosity, refractive index, and surface tension of binary liquid mixtures of diacetone alcohol with benzene, chlorobenzene, and bromobenzene with less than 1% error. Baroutain et al. (2008) applied ANN to estimate the density of pure oil-based methylester biodiesel and the model gave an average absolute percentage deviation of 0.29% for its prediction. Pouryousefi et al. (2016) used
backpropagation neural networks (BPNN) and radial basis neural work (RBFNN) to predict the density, viscosity, heat capacity, refractive index, thermal conductivity and thermal diffusivity of aqueous quaternary systems of amine solvents. The results showed that the neural network models were capable of predicting the physical properties with less than 1% of average absolute deviation. The developed neural network models were used to predict the composition of the amine solvent given the physical properties and the predictions given by the BPNN and RBFNN showed a deviation of 4.58% and 5.75%, respectively, when compared to the experimental results (Pouryousefi, 2014).

9.2 The Dataset

The EURECA algorithm was applied to the experimental results obtained from (Pouryousefi et al. 2016; Pouryousefi, 2014). Pouryousefi et al. (2016) measured the physical and heat transport properties of monoethanolamine (MEA) solutions that were blended with different ratios of carbon dioxide (CO₂), methyldiethanolamine (MDEA) and water at different temperatures. The measured properties include density, viscosity, refractive index, heat capacity, thermal conductivity, and thermal diffusivity. The base solution consisted of 5 M aqueous MEA solution and 1.5 M MDEA; the base solution was then mixed with various amounts of CO₂, ranging between 0 – 0.6 mol CO₂/mol amine. The properties of density, viscosity and refractive index were measured in the temperature range of 298.15 K to 333.15 K, and heat capacity, thermal conductivity and thermal diffusivity were measured in the temperature range of 298.15 K to 278.15 K. A total of 144 data tuples were collected for the attribute of viscosity while 174 tuples were collected for all the other properties. There was no missing or noisy value in the dataset.
and therefore no filtering was required. The predictor variables of the trained ANN models in (Pouryousefi et al., 2016) include the mole fraction of each component of the amine solvent of CO$_2$, MEA, MDEA and water, and the temperature the property was measured at, and the predicted variable is each of the specified physical properties, including density, viscosity, refractive index, heat capacity, thermal conductivity, and thermal diffusivity.

### 9.3 Application of the EURECA Algorithm

Before training the ANN models using Rapidminer’s (trademark of Rapidminer) neural network algorithm, the values of the dataset were first normalized to the range of [-1, 1]. The neural network algorithm provided by Rapidminer uses the sigmoid function as the activation function in the hidden layer and the linear function in the output layer. During training of the ANN models, different settings of the training cycle, learning rate and momentum, and different number of hidden nodes were used to ensure satisfactory accuracies of the models can be achieved. Except for the parameters of heat capacity and thermal conductivity, an ANN model was trained for each of the other physical properties of the amine solvent. Each model has five predictor variables and one predicted variable. The predictor variables of the ANN and EURECA models include the mole fraction of each component of the amine solvent of CO$_2$, MEA, MDEA and water, and the temperature the property was measured at, and the predicted variable is each of the specified physical properties, including density, viscosity, refractive index, heat capacity, thermal conductivity, and thermal diffusivity.
After the ANN model was developed, the EURECA algorithm was applied to the trained ANN models. The values of \((-2.49, 2.49)\) were used as the initial breakpoints and were adjusted accordingly to improve the model’s accuracy. Even though the dataset used for training the ANN model had been normalized to \([-1, 1]\), the rules or equations given by the EURECA algorithm are in terms of the real values of the dataset.

The predictive accuracies of the trained ANN models and the generated EURECA models were measured in terms of \(R^2\) and mean square error (MSE). \(R^2\) is called the coefficient of determination and it represents how close the fitted line is to the actual values. The closer the value of \(R^2\) to 1, the more accurately the fitted line models the actual values. The MSE measures the deviation of the predicted values from the actual values. The lower the value of MSE, the lower the deviation of the predicted values from the actual values. The values of \(R^2\) and MSE can be calculated:

\[
R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}
\]

\[
MSE = \frac{\sum_i (y_i - \hat{y}_i)^2}{n}
\]

where \(y_i\) = the expected output of the \(i^{th}\) dataset, \(\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i\) and \(\hat{y}\) = predicted output and \(n\) = number of data tuples in the dataset. The results given by the trained ANN models and the generated EURECA models are summarized in Table 9.1.
Table 9.1 – Accuracies of the trained ANN models and the generated EURECA models

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ANN</th>
<th>EURECA</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$R^2$</td>
<td>MSE</td>
<td>No. of hidden nodes</td>
<td>$R^2$</td>
<td>MSE</td>
</tr>
<tr>
<td>1. Density</td>
<td>0.9971</td>
<td>0.00104</td>
<td>4</td>
<td>0.9994</td>
<td>0.00000</td>
</tr>
<tr>
<td>2. Viscosity</td>
<td>0.9852</td>
<td>0.00204</td>
<td>4</td>
<td>0.9852</td>
<td>0.05660</td>
</tr>
<tr>
<td>3. Heat Capacity (1)</td>
<td>0.8919</td>
<td>0.01949</td>
<td>5</td>
<td>0.9928</td>
<td>0.00047</td>
</tr>
<tr>
<td>4. Heat Capacity (2)</td>
<td>0.9078</td>
<td>0.01614</td>
<td>6</td>
<td>0.9895</td>
<td>0.00068</td>
</tr>
<tr>
<td>5. Refractive Index</td>
<td>0.9868</td>
<td>0.00252</td>
<td>4</td>
<td>0.9970</td>
<td>0.00000</td>
</tr>
<tr>
<td>6. Thermal Conductivity</td>
<td>0.9631</td>
<td>0.00509</td>
<td>4</td>
<td>0.9741</td>
<td>16.68630</td>
</tr>
<tr>
<td>7. Thermal Diffusivity (1)</td>
<td>0.9030</td>
<td>0.01360</td>
<td>4</td>
<td>0.9225</td>
<td>0.00873</td>
</tr>
<tr>
<td>8. Thermal Diffusivity (2)</td>
<td>0.9232</td>
<td>0.01078</td>
<td>6</td>
<td>0.9405</td>
<td>0.00670</td>
</tr>
</tbody>
</table>
The values of MSE cannot be used as comparison between the trained ANN models and the EURECA models because the MSE values of the trained ANN models were calculated using the normalized values whereas the MSE values of the EURECA models were calculated using the real values of the predicted attribute. Therefore, the value of $R^2$ was used for comparison. As shown by the $R^2$ values in Table 9.1, all the generated EURECA models have a higher predictive or at least equivalent accuracy compared to the trained ANN models. It can be seen that the EURECA models maintained the high accuracies given by the trained ANN models of the density (Dataset #1), viscosity (Dataset #2) and refractive index (Dataset #5) datasets; the $R^2$ values of the EURECA models of these datasets are 0.9994, 0.9852 and 0.9970, respectively. The EURECA algorithm also improved the accuracy of the heat capacity models (Dataset #3 and #4) significantly from 0.8919 for Dataset #3 and 0.9078 for Dataset #4, to 0.9928 and 0.9895, respectively. In contrast, the accuracies of the thermal diffusivity models were only improved slightly by the EURECA algorithm. The $R^2$ values of the two generated EURECA models for thermal diffusivity (Dataset #7 and #8) are 0.9225 and 0.9405, compared to the $R^2$ values of the trained ANN models, which are 0.9030 and 0.9232, respectively. One possible explanation on the relatively less significant improvement for the thermal diffusivity datasets is that the predictor and predictive variables are in a more complex relationship than the other datasets. The dataset may require an ANN model with two hidden layers, or different activation functions. Nonetheless, a $R^2$ value of 0.9225 and 0.9405 indicate reasonable accuracies of the EURECA models.
The results in Table 9.1 also revealed that the number of derived equations increased when the number of hidden nodes was increased. The EURECA algorithm generated models with ten equations or less for all the trained ANN models with four hidden nodes; whereas 15 rules were produced for the heat capacity model with 5 hidden nodes, and 17 and 20 rules were produced for the two models with 6 hidden nodes. This relationship exists because the number of hidden nodes directly affects the number of clusters a dataset can have and since one rule or equation is associated with each cluster, the number of hidden nodes also affects the number of rules generated. Assuming there are \( j \) hidden nodes, and each node is divided into three groups, i.e. two breakpoints, then the maximum number of clusters a given dataset can have is \( 3^j \). Therefore, a higher \( j \) value means more clusters. In other words, there is a trade-off between the accuracies of the ANN and EURECA models and the comprehensibility of the extracted rule sets.

9.4 Results and Analysis

9.4.1 Knowledge Extracted from the Generated EURECA Models

Each of the rules in a EURECA model covers a specific part of the problem space. Knowledge about the problem domain can be extracted by examining the coverage of each rule and the coefficients associated with the parameters in the equations. A rule that has a higher coverage indicates it is a more significant rule in the problem domain. As well, the magnitudes of the predictor attributes’ coefficients reflect the relative significance of the predictor attributes on the predicted attribute. In section 9.4.1.1, the Density dataset (Dataset #1) is used as a detail illustration on the knowledge extraction process. Then, the discussion presented a comparison of the knowledge that can be
9.4.1.1 Detail Illustrate with the Density Dataset

As shown in Table 9.1, a total of 9 rules were extracted for the Density dataset. Rule #6 had the highest coverage and covered 52 out of the 174 of the data tuples. As the most significant rule for the dataset, Rule #6 stated that

\[
\text{IF}\quad ((0.0168 \leq M_{CO2} \leq 0.0858) \, \text{AND} \, (0.1249 \leq M_{MEA} \leq 0.1342) \, \text{AND} \, (0.0250 \leq M_{MDEA} \leq 0.0396) \, \text{AND} \, (0.7460 \leq M_{H2O} \leq 0.8162) \, \text{AND} \, (298 \leq T \leq 323))
\]

\[
\text{THEN}\quad \text{Density} = -480.90 + 483.51 \, M_{CO2} + 451.04 \, M_{MEA} + 509.13 \, M_{MDEA} + 486.11 \, M_{H2O} - 6.4\times10^{-4} \, T
\]

where $M_{CO2}$, $M_{MEA}$, $M_{MDEA}$ and $M_{H2O}$ are the molar fraction of CO$_2$, MEA, MDEA and water, respectively, and $T$ is temperature in Kelvin (K).

From the above rule, it can be observed that the mole fraction of MDEA had the largest coefficient and hence was the most dominating predictor attribute in the equation. The complete set of rules given by the EURECA algorithm along with the coverage of each rule is shown in Table 9.2 and this rule set is called Rule Set A. By examining the predictor attributes’ coefficients in all the equations, it can be observed that $M_{MDEA}$ is associated with the largest coefficient in the three rules with the highest coverage, i.e. Rule #1, #4, and #6, which together cover a total of 122 data tuples. Hence, $M_{MDEA}$ was
considered as the most significant attribute in this problem. Following the same logic, the second most significant attribute was $M_{CO2}$ which had the largest coefficient in 4 generated rules, i.e. Rule #2, #5, #7, #8, which together had a total coverage of 46 data tuples, and the third most significant attribute was $M_{MEA}$ which covers 36 data tuples.
Table 9.2 – Rules extracted by the EURECA algorithm (Rule Set A) for the Density dataset

<table>
<thead>
<tr>
<th>Rule</th>
<th>Coverage</th>
<th>IF…</th>
<th>THEN …</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>34</td>
<td>$(0.0000 \leq \text{M}<em>{\text{CO}<em>2} \leq 0.0632)$ AND $(0.1149 \leq \text{M}</em>{\text{MEA}} \leq 0.1220)$ AND $(0.0115 \leq \text{M}</em>{\text{MDEA}} \leq 0.0120)$ AND $(0.8103 \leq \text{M}_{\text{H}_2\text{O}} \leq 0.8660)$ AND $(298 \leq T \leq 323)$</td>
<td>Density = $53.85 - 50.71 \text{M}<em>{\text{CO}<em>2} - 1486.08 \text{M}</em>{\text{MEA}} + 4207.74 \text{M}</em>{\text{MDEA}} + 90.20 \text{M}_{\text{H}_2\text{O}} - 4.7E-04 T$</td>
</tr>
<tr>
<td>#2</td>
<td>20</td>
<td>$(0.0000 \leq \text{M}<em>{\text{CO}<em>2} \leq 0.0308)$ AND $(0.1275 \leq \text{M}</em>{\text{MEA}} \leq 0.1360)$ AND $(0.0255 \leq \text{M}</em>{\text{MDEA}} \leq 0.0340)$ AND $(0.8162 \leq \text{M}_{\text{H}_2\text{O}} \leq 0.8380)$ AND $(298 \leq T \leq 323)$</td>
<td>Density = $1.13 + 1.13 \text{M}_{\text{CO}<em>2} + 0.50 \text{M}</em>{\text{MEA}} - 5.9E-04 T$</td>
</tr>
<tr>
<td>#3</td>
<td>2</td>
<td>$\text{M}<em>{\text{CO}<em>2} = 0.0000$ AND $\text{M}</em>{\text{MEA}} = 0.1360$ AND $\text{M}</em>{\text{MDEA}} = 0.0340$ AND $\text{M}_{\text{H}_2\text{O}} = 0.8300$ AND $T = 298$</td>
<td>Density = 1.03</td>
</tr>
<tr>
<td>#4</td>
<td>36</td>
<td>$(0.0000 \leq \text{M}<em>{\text{CO}<em>2} \leq 0.0610)$ AND $(0.1410 \leq \text{M}</em>{\text{MEA}} \leq 0.1550)$ AND $(0.0420 \leq \text{M}</em>{\text{MDEA}} \leq 0.1550)$ AND $(0.6900 \leq \text{M}_{\text{H}_2\text{O}} \leq 0.8170)$ AND $(298 \leq T \leq 323)$</td>
<td>Density = $267.16 - 264.54 \text{M}<em>{\text{CO}<em>2} - 265.69 \text{M}</em>{\text{MEA}} - 265.99 \text{M}</em>{\text{MDEA}} - 265.99 \text{M}_{\text{H}_2\text{O}} - 6.3E-04 T$</td>
</tr>
<tr>
<td>#5</td>
<td>8</td>
<td>$(0.0632 \leq \text{M}<em>{\text{CO}<em>2} \leq 0.0772)$ AND $(0.1149 \leq \text{M}</em>{\text{MEA}} \leq 0.1214)$ AND $(0.0115 \leq \text{M}</em>{\text{MDEA}} \leq 0.0243)$ AND $(0.7771 \leq \text{M}_{\text{H}_2\text{O}} \leq 0.8103)$ AND $(298 \leq T \leq 323)$</td>
<td>Density = $1.31 - 0.31 \text{M}_{\text{CO}_2} - 5.5E-04 T$</td>
</tr>
<tr>
<td>#6</td>
<td>52</td>
<td>$(0.0168 \leq \text{M}<em>{\text{CO}<em>2} \leq 0.0858)$ AND $(0.1249 \leq \text{M}</em>{\text{MEA}} \leq 0.1342)$ AND $(0.0250 \leq \text{M}</em>{\text{MDEA}} \leq 0.0396)$ AND $(0.7460 \leq \text{M}_{\text{H}_2\text{O}} \leq 0.8162)$ AND $(298 \leq T \leq 323)$</td>
<td>Density = $-480.90 + 483.51 \text{M}<em>{\text{CO}<em>2} + 451.04 \text{M}</em>{\text{MEA}} + 509.13 \text{M}</em>{\text{MDEA}} + 486.11 \text{M}_{\text{H}_2\text{O}} - 6.4E-04 T$</td>
</tr>
<tr>
<td>#7</td>
<td>9</td>
<td>$(0.0303 \leq \text{M}<em>{\text{CO}<em>2} \leq 0.0852)$ AND $(0.1372 \leq \text{M}</em>{\text{MEA}} \leq 0.1415)$ AND $(0.0412 \leq \text{M}</em>{\text{MDEA}} \leq 0.0566)$ AND $(0.7168 \leq \text{M}_{\text{H}_2\text{O}} \leq 0.7913)$ AND $(298 \leq T \leq 323)$</td>
<td>Density = $1.20 + 1.52 \text{M}_{\text{CO}_2} - 5.9E-04 T$</td>
</tr>
<tr>
<td>#8</td>
<td>9</td>
<td>$(0.0852 \leq \text{M}<em>{\text{CO}<em>2} \leq 0.0977)$ AND $(0.1395 \leq \text{M}</em>{\text{MEA}} \leq 0.1415)$ AND $(0.0558 \leq \text{M}</em>{\text{MDEA}} \leq 0.0566)$ AND $(0.7070 \leq \text{M}_{\text{H}_2\text{O}} \leq 0.7168)$ AND</td>
<td>Density = $1.26 + 0.83 \text{M}_{\text{CO}_2} - 5.9E-0.4 T$</td>
</tr>
<tr>
<td>#9</td>
<td>4</td>
<td>( M_{\text{CO}<em>2} = 0.0000 ) AND ( M</em>{\text{MEA}} = 0.1360 ) AND ( M_{\text{MDEA}} = 0.0340 ) AND ( M_{\text{H}_2\text{O}} = 0.8300 ) AND ( (318 \leq T \leq 323) )</td>
<td>Density = 1.04 – 9.2E-0.5 ( T )</td>
</tr>
</tbody>
</table>
In order to simplify the generated rule set and to remove less significant rules, the decision tree algorithm described in (Therneau et al., 1997) was employed to classify the data tuples into the rules they were assigned to by EURECA. In other words, the predictor attributes for the decision tree model were $M_{\text{CO}_2}$, $M_{\text{MEA}}$, $M_{\text{MDEA}}$, $M_{\text{H}_2\text{O}}$ and $T$, and the predicted attribute was the rules assigned by the EURECA algorithm. The classification tree was implemented using the \texttt{rpart()} function in R, and details of the algorithm can be found in (Therneau et al., 1997). The generated classification tree is shown in Figure 9.1.

As shown in Figure 9.1, the generated decision tree contains only Rule #1, #2, #4, #6 and #7 as the classifier of the dataset. Starting from the far left branch of the classification tree, the rules given by the decision tree algorithm (called Rule set B) are as follows:

1. IF ($M_{\text{MDEA}} < 0.018$) THEN Density is given by Rule #1:
   \[
   \text{Density} = 53.85 - 50.71 M_{\text{CO}_2} - 1486.08 M_{\text{MEA}} + 4207.74 M_{\text{MDEA}} + 90.20 M_{\text{H}_2\text{O}} - 4.7 E^{-04} T 
   \]

2. IF ((\(M_{\text{MDEA}} \geq 0.018\)) AND (\(M_{\text{MEA}} \geq 0.14\))) THEN Density is given by Rule #4:
   \[
   \text{Density} = 267.16 - 264.54 M_{\text{CO}_2} - 265.69 M_{\text{MEA}} - 265.99 M_{\text{MDEA}} - 265.99 M_{\text{H}_2\text{O}} - 6.3 E^{-04} T 
   \]

3. IF ((\(M_{\text{MDEA}} \geq 0.018\)) AND (\(M_{\text{MDEA}} \geq 0.14\)) AND (\(M_{\text{H}_2\text{O}} \geq 0.82\))) THEN Density is given by Rule #2:
   \[
   \text{Density} = 1.13 + 1.13 M_{\text{CO}_2} + 0.50 M_{\text{MEA}} - 5.9 E^{-04} T 
   \]

4. IF ((\(M_{\text{MDEA}} \geq 0.018\)) AND (\(M_{\text{MDEA}} < 0.14\)) AND (\(M_{\text{H}_2\text{O}} < 0.82\))) THEN Density is given by Rule #6:
Density = -480.90 + 483.51M_{CO2} + 451.04M_{MEA} + 509.13M_{MDEA} + 486.11M_{H2O} – 6.4E-04T

5. \text{IF((}M_{MDEA} \geq 0.018) \text{ AND (}M_{MDEA} \geq 0.14) \text{ AND (}M_{H2O} < 0.82)) \text{ THEN Density is given by Rule #7:}

\text{Density} = 1.20 + 1.52M_{CO2} – 5.9E-04T

It can be seen that the decision tree algorithm gave a refined set of rules, which has a reduced the size from 9 rules to 5 rules, and fewer conditions in the IF-statement of the rule. While the Rule Set A given by the EURECA algorithm each had 5 conditions in the IF-statement as shown in Table 9.2, the refined set of rules (called Rule Set B) had at most 3 conditions in the IF-statement. With fewer conditionals and improved comprehensibility, however, the predictive accuracy of Rule Set B was slightly lower than that of Rule Set A. Rule Set B had a MSE of 1.03E-05 and an R^2 value of 0.9956, whereas Rule Set A had a MSE of 1.46E-06 and an R^2 value of 0.9994. By examining the magnitude of the coefficients associated with the predictor attribute in the classification tree rules, the ranking of the attributes’ significance was \( M_{MDEA}, M_{MEA}, \) and \( M_{CO2} \). The rankings of the significant attributes given by Rule Set A and Rule Set B are summarized in Table 9.3.
Figure 9.1 – Classification tree obtained for the Density dataset using the EURECA extracted rules
Table 9.3 – Ranking of attributes’ significance given by Rule Set A and Rule Set B

<table>
<thead>
<tr>
<th>Significance Ranking</th>
<th>Rule Set A</th>
<th>Rule Set B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1&lt;sup&gt;st&lt;/sup&gt;</td>
<td>$M_{MDEA}$</td>
<td>$M_{MDEA}$</td>
</tr>
<tr>
<td>2&lt;sup&gt;nd&lt;/sup&gt;</td>
<td>$M_{CO2}$</td>
<td>$M_{MEA}$</td>
</tr>
<tr>
<td>3&lt;sup&gt;rd&lt;/sup&gt;</td>
<td>$M_{MEA}$</td>
<td>$M_{CO2}$</td>
</tr>
</tbody>
</table>
The results were showed to the domain expert for validation. The domain expert compared the results to the semiempirical correlation of Weiland and nonadditive equations for amine solution density (Weiland et al., 1998), as shown in eqs (9.1) – (9.3):

\[
\rho = \sum_{i=1}^{\text{R}} x_i \rho_i \quad (9.1)
\]

\[
\rho = \frac{\sum_{i=1}^{\text{R}} (x_i M_i)}{v} \quad (9.2)
\]

Where

\[
V = x_{AM} \cdot v_{AM} + x_{H2O} \cdot v_{H2O} + x_{CO2} \cdot v_{CO2} + x_{AM} \cdot x_{H2O} \cdot v^* + x_{AM} \cdot x_{CO2} \cdot v^{**} \quad (9.3)
\]

\(x_i\) = mole fraction of component \(i\),

\(\rho_i\) = density of component \(i\),

\(M_i\) = molecular weight of component \(i\),

\(v_i\) = partial molar volume of component \(i\),

\(v^*\) = partial molar volume associated with the interaction between amine and water

\(v^{**}\) = partial molar volume associated with the interaction between amine and CO\(_2\)

\(V\) = total volume of mixture, and AM = amine solvent

According to equation (9.2), the density of the solution is determined by the sum of the product between the mole fraction and molecular weight of the components. This means the value of density is dominated by the component with the heaviest molecular weight. Since MDEA has the heaviest molecular weight of 119.17 g/mol among the four component of MDEA, MEA, CO\(_2\) and water of the mixed solvent, it is reasonable that the generated rules identify the predictor attribute \(M_{\text{MDEA}}\) as the most significant attribute. For similar reasons, since MEA is the second heaviest molecule with a molecular weight
of 61.09 g/mol, and should be identified as the second most significant predictor attribute. However, the generated EURECA rule set identified $M_{CO2}$ as second most significant attribute and $M_{MEA}$ as the third most significant attribute. The domain expert explained that since CO$_2$ reacted with amine and water in the mixture, the CO$_2$ considered was no longer pure CO$_2$. The mixed CO$_2$ could become heavier than MEA and thus $M_{CO2}$ could become the second most significant attribute. The domain expert concluded that the results given by the EURECA rule sets were reasonable because the exact state of CO$_2$ was uncertain.

9.4.1.2 Verification with Experimental Observations

In the experimental work performed in Pouryousefi (2014), the author analyzed only the effect of CO$_2$ and temperature on the chemical and physical properties of density, viscosity, refractive index, heat capacity, thermal conductivity, and thermal diffusivity of the amine solvent. The author kept the concentration of MEA and MDEA at constant and observed the effect on the chemical and physical properties of loading the mixture with different amount of CO$_2$ on the physical properties. She observed that density, viscosity and refractive index were all affected by CO$_2$ positively, i.e. the density, viscosity and refractive index of the mixture increased when the mole fraction of CO$_2$ increased. In contrast, heat capacity, thermal conductivity and thermal diffusivity were affected by the amount of CO$_2$ negatively. In other words, these values decreased when the amount of CO$_2$ increased. Pouryousefi (2014) suggested that, apart from heat capacity, the values of all the other properties decreased as temperature was increased. These observations of Pouryousefi (2014) were compared to the information revealed by the EURECA models.
based on the most significant rule in each of the models, and the results are summarized in Table 9.4. “+ve” denotes a positive relationship between the attribute and the property, and “-ve” means a negative relationship.

It can be seen in Table 9.4 that apart from refractive index, the observations made by examining the coefficients of the generated equations agree with those made through experiment in Pouryousefi (2014). This case study demonstrates that EURECA can generate equations that are statistically accurate; further involvement of an expert can validate the significance of the equations in the physical domain.
Table 9.4 – Observations made by Pouryousefi (2014) Vs the EURECA models

<table>
<thead>
<tr>
<th>Properties</th>
<th>Pouryousefi (2014)</th>
<th>EURECA Models</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CO₂</td>
<td>Temperature</td>
</tr>
<tr>
<td>Density</td>
<td>+ve</td>
<td>-ve</td>
</tr>
<tr>
<td>Viscosity</td>
<td>+ve</td>
<td>-ve</td>
</tr>
<tr>
<td>Heat Capacity (1)</td>
<td>-ve</td>
<td>+ve</td>
</tr>
<tr>
<td>Heat Capacity (2)</td>
<td>-ve</td>
<td>+ve</td>
</tr>
<tr>
<td>Refractive Index</td>
<td>+ve</td>
<td>-ve</td>
</tr>
<tr>
<td>Thermal Conductivity</td>
<td>-ve</td>
<td>-ve</td>
</tr>
<tr>
<td>Thermal Diffusivity</td>
<td>-ve</td>
<td>-ve</td>
</tr>
<tr>
<td></td>
<td>(1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(2)</td>
<td></td>
</tr>
</tbody>
</table>
9.4.2 Monitoring the Amine Solvent using the Generated EURECA Models

In addition to enhancing understanding of a studied domain, the extracted rules can be used to solve the back calculation problem encountered while monitoring the state of amine in the CO₂ capture process. As explained earlier, the back calculation problem arises when there is a target predicted value one wishes to achieve but sufficient information on the values of the predictor attributes is lacking. In the case of CO₂ process system monitoring, the known target predicted value is the measured solvent properties, such as density, viscosity and refractive index, and the only known predictor attribute is the temperature. The values of the remaining four predictor attributes, i.e. mole fraction of CO₂, MEA, MDEA and water, are unknown, and needed be determined. To solve the back calculation problem using the extracted rules, the following approach was adopted:

(1) Identify the matching equations for the given known values

Since each of the equations covers a specific range of values of the predictor and predicted variables, the number of equations can be reduced to the ones that cover the given known values. For a given scenario, more than one equation may be identified.

(2) For each of the matching equations:

a. Identify the ranges of values of the unknown parameters covered by the equation and list the values of between the ranges in an increment of 0.001 for each unknown. For example, consider the range of values [0.063, 0.077] for the mole fraction of CO₂, then the list of values for the mole fraction of CO₂ in an increment of 0.001 is (0.063, 0.064, 0.065, 0.066, 0.067, 0.068, 0.069, 0.070, 0.071, 0.072, 0.073, 0.074, 0.075, 0.076, 0.077). Since the unknown attributes are mole fractions of the component
in the mixed solvent, their values are always less than 1. Therefore the increment of 0.001 was chosen such that it is small enough to account for the possible combinations of mole fractions.

b. Test all value combinations of the unknown attributes given by the lists of values using the matching equation.

c. The combination that returns a calculated predictor value closest to the desired target predictor value is selected as the solution.

(3) Repeat step (2) for all matching equations.

(4) When there are more than one matching equation, the equation and combination that return a calculated predictor value closest to the desired target predictor value is selected as the solution.

Since the EURECA models for the datasets density, heat capacity(1) and refractive index had the highest accuracy, the back calculation problem were solved using these three models. The scenarios of two, three and four unknowns were tested and twenty data tuples were randomly selected from the dataset for each scenario as the test cases. The unknown of predictor attributes were also randomly selected in each test cases. In order to evaluate the performance of the EURECA models on the back calculation problems, the following values were examined:

(1) The average absolute deviation of the calculated predicted attribute values, $E_{y0}$, this measures the difference between the calculated value and the target or actual value of the predicted or output attribute, i.e. error.
(2) The average absolute deviation of the calculated predictor attributes values, $E_{x0}$, this measures the difference between the calculated value and the target or actual value of the predictor or input attribute, i.e. error.

The average absolute deviation (AAD) is given by:

$$AAD \, (\%) = \frac{\text{Absolute} \left( \frac{\text{Target Value} - \text{Calculated Value}}{\text{Target Value}} \right)}{} \times 100\%$$

The average absolute deviations of the calculated predicted attribute values and the target predicted given by the EURECA models for the density, heat capacity (1) and refractive index are summarized in Table 9.5 and the plot of average AAD error between calculated and target predicted attribute Vs number of unknown attributes is shown in Figure 9.2. When there are more than one matching equations in a particular problem scenario, more than one solution will be given, and the solution that gives the lowest AAD error between the calculated and target predicted attribute was chosen to be the solution. As shown in Table 9.5 and Figure 9.2, the average absolute deviation between the calculated and target predicted attribute decreases as the number of unknown attribute increases. On average, the AAD between the calculated and target predicted variable when there were two unknown attributes was 1.683%. The AAD decreased to 0.089% when there were three unknowns and it was almost 0% when there were four unknown attributes.
Table 9.5 – The AAD between the calculated and target predicted attribute given by the Density, Heat Capacity (1) and Refractive Index EURECA models.

<table>
<thead>
<tr>
<th>Number of Unknown</th>
<th>Density</th>
<th>Heat Capacity</th>
<th>Refractive Index</th>
<th>$E_{y0}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.619582</td>
<td>3.420228</td>
<td>0.008824</td>
<td>1.682878</td>
</tr>
<tr>
<td>3</td>
<td>0.068446</td>
<td>0.196091</td>
<td>0.001756</td>
<td>0.088764</td>
</tr>
<tr>
<td>4</td>
<td>0.000044</td>
<td>0.001073</td>
<td>0.000035</td>
<td>0.000384</td>
</tr>
</tbody>
</table>

Figure 9.2 – AAD between the calculated and target predicted attribute Vs number of unknown variables.
The average absolution deviation between the calculated and actual unknown predictor variables, i.e. the mole fractions of the amine solvent composition, are summarized in Table 9.6 and the plot of AAD error between the calculated and actual unknown predictor variables Vs number of unknowns is shown in Figure 9.3. As shown in Table 9.6, the average AAD error between the calculated and actual predictor attributes increases as the number of unknown attributes increases. On average the calculated predictor attributes has an error of 13.66 % when there were two unknown attributes. When there were three unknown attributes, the error increased to 24.76%; and when there were four unknown attributes, the error was 44.06%.

Although an AAD of 44.06% seems high, which implies the calculated values were inaccurate, the majority of the calculated values were only between 10 – 30 % error rates. Figure 9.4 shows the distribution of the solved unknown attributes in terms of AAD percentage error given by the density EURECA model when there were four unknowns. The x-axis of the plot in Figure 9.4 shows the bins of the AAD of the solved values and the y-axis shows the percentage of the solved attributes that falls into the bins shown in the x-axis. As it can be seen in Figure 9.4, 46.25% of the calculated attributes falls within the 10 – 20% AAD percentage error 22.5% falls with 20 – 30 %. This accounts for 68.75% of the total calculated unknown predictor attributes. Similar distributions were observed from the results given by the EURECA models for the heat capacity (1) and refractive index datasets.
Table 9.6 – The AAD between the solved and actual predictor attributes given by the Density, Heat Capacity (I) and Refractive Index EURECA models.

<table>
<thead>
<tr>
<th>Number of Unknown</th>
<th>Density</th>
<th>Heat Capacity</th>
<th>Refractive Index</th>
<th>$E_{x0}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>8.57</td>
<td>9.92</td>
<td>22.49</td>
<td>13.66</td>
</tr>
<tr>
<td>3</td>
<td>13.52</td>
<td>47.23</td>
<td>13.55</td>
<td>24.76</td>
</tr>
<tr>
<td>4</td>
<td>36.46</td>
<td>67.99</td>
<td>27.72</td>
<td>44.06</td>
</tr>
</tbody>
</table>

Figure 9.3 – AAD between the solved and actual predictor attributes Vs number of unknown variables
Figure 9.4 – AAD percentage error distribution of the solved unknown attributes given by the density ERUECA model
The results for the back calculation scenarios were presented to the domain expert, who considered that an error rate of 30% to be acceptable in the problem domain. He confirmed that with almost 70% of the calculated values falling within 30% error rate, the back calculation solutions given by the EURECA models can be considered as providing reasonable guidelines to operators who monitor the amine solvent ratio during the CO₂ capture system operation.

It is important to note that no matching equations were found for three test cases using the heat capacity (1) and refractive index EURECA models when there were two unknowns, and no matching equations were found for one test cases using the capacity (1) and refractive index EURECA models when there were three unknowns. This suggests that the EURECA models of heat capacity (1) and refractive index are not as reliable as the density EURECA model which provides at least one solution for all the given test cases.

In Pouryousefi’s study (2014), two types of ANN models with the backpropagation feedforward algorithm (BPNN) and radial basis function (RBFNN) were trained to predict the composition of the amine solvent components by using the physical properties as inputs and the results were compared to the composition measured by laboratory equipments. The results showed the RBFNN models had an average absolute deviation (AAD) of 2.49%, 4.58% and 2.97% for the composition of MEA, MDEA and CO₂, respectively, and the BPNN models had an AAD of 3.00%, 5.75% and 5.02% for the composition of MEA, MDEA and CO₂, respectively. The composition of water was not
examined in her study. These results were compared to the AAD given by the density rule set when one unknown predictor attribute was present and they are summarized in Table 9.7. It can be seen from Table 9.7 that the EURECA density model was able to determine the composition of the amine solvent more accurately than the ANN models presented in (Pouryousefi, 2014). The AAD given by the EURECA density model was 0.19%, 3.90% and 0.32% for MEA, MDEA and CO₂, respectively. While the ANN models presented in Pouryousefi (2014) did not examine the composition of water, the EURECA Density model could determine the composition of water with 1.43% AAD. It can be seen from Table 9.7 that, comparing with the ANN modeling results, the EURECA algorithm was able to generate more accurate results using the approach of back calculation proposed in this study.
Table 9.7 – The AAD of individual components given by the RBFNN and BPNN from (Pouryousefi, 2014) and the EURECA Density model

<table>
<thead>
<tr>
<th>Component</th>
<th>RBFNN</th>
<th>BPNN</th>
<th>EURECA Density rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEA</td>
<td>2.49%</td>
<td>3.00%</td>
<td>0.19%</td>
</tr>
<tr>
<td>MDEA</td>
<td>4.58%</td>
<td>5.75%</td>
<td>3.90%</td>
</tr>
<tr>
<td>CO₂</td>
<td>5.02%</td>
<td>5.02%</td>
<td>0.32%</td>
</tr>
<tr>
<td>WATER</td>
<td>N/A</td>
<td>N/A</td>
<td>1.43%</td>
</tr>
</tbody>
</table>
9.5 Conclusion

This case study presents the application of the EURECA algorithm for solving operational problems related to amine solvent monitoring in CO₂ capture process system. The amine solvent dataset examined in this study was obtained through experimental work that involves a quaternary system consisting of the four components of carbon dioxide, water, methyldiethanolamine (MDEA) and monoethanolamine (MEA) (Pouryousefi, 2014). The studied physical properties of the amine solvent include density, viscosity, heat capacity, refractive index, thermal conductivity and thermal diffusivity. The ANN models were trained to model the relationships among each of the physical properties and the mole fraction of each of the four solvent components at a given temperature. An ANN model was trained for each of the physical properties of density, viscosity, refractive index and thermal conductivity. All the trained ANN models had a $R^2$ value of greater than 0.96. Two ANN models with different numbers of hidden neurons were trained for heat capacity and thermal diffusivity and by increasing the number of hidden neurons, the accuracies in terms of $R^2$ of the ANN models for heat capacity and thermal diffusivity was improved from 0.8919 to 0.9078 and 0.9030 to 0.9232, respectively. The generated EURECA models all had a higher predictive accuracy compared to their underlying ANN models. The predictive accuracies of the heat capacity and thermal diffusivity models were improved from 0.9078 to 0.9895 and from 0.9232 to 0.9405, respectively.

By examining the coverage of the extracted rules or equations, the most dominating rule of each physical property was identified. The most significant predictor attributes for
each physical property was revealed by examining the magnitude of the coefficients. The Density dataset was used to illustrate the process of knowledge extraction, and by employing the decision tree algorithm, the extracted rule sets could be further refined. The EURECA rules identified $M_{\text{MDEA}}$ as the most significant followed by $M_{\text{CO}_2}$ and $M_{\text{MEA}}$. The results were verified by the domain expert as correct. The relationship between the physical properties and the mole fraction of each of the solvent component was validated against the experimental observations in (Pouryousefi, 2014).

In addition to providing explicit relationships between the predictor and predicted variables, the extracted rules or equations can be applied to solve for unknown parameter values in the back calculation problem. A sample scenario involves identifying the concentration of each solvent component given the values of the physical properties and the temperature at which the properties were measured at. This scenario was examined using the generated EURECA models for the datasets of density, heat capacity (1) and refractive index. The results showed that almost 70% of the calculated values have an average absolute deviation of less than 30% when four unknown was presented. The solutions generated were validated by the domain expert, who confirmed that an error rate of under 30% is considered acceptable in the problem domain. The rules generated from the application of EURECA were considered as providing reasonable guidelines for determining the concentration of amine solvent during operation of the CO$_2$ capture process system. By comparing the results to the ANN models presented in (Pouryousefi, 2014), it was shown that the generated EURECA models could determine the concentration of each solvent component more accurately than the ANN models.
CHAPTER 10 CONCLUSION AND FUTURE WORK

10.1 Conclusions

In this study, three artificial neural network (ANN) rule extraction algorithms, namely the PWL, the enhanced-PWL-ANN and the EURECA algorithms, were proposed. The major contributes of this study can be summarized as below:

1. The proposed ANN rule extraction algorithms generate explicit relationships between the predictor and predicted variables in the form of multiple linear regression equations so that a better understanding of the studied problem can be gained. This opens up the “black-box” of the ANN model. By examining the correlation between the assigned weight values and the values of the predictor attributes, and understanding about the learning process of the ANN algorithm is obtained. It was discovered that there was a correlation between the assigned weights of the input data tuples and the hypothesized linear relationship that exists among subsets of data in a given problem domain.

2. The EURECA algorithm proposed adopt a novel approach of clustering the data tuples according to their weights to the hidden neurons.

3. The three case studies demonstrated that the developed EURECA algorithm was suitable for modeling real industrial engineering problems, and by applying the EURECA algorithm, valid and comprehensible rules could be generated to enhance the domain experts’ understanding of the studied problem. The EURECA algorithm can also be used for solving the back calculation problems.

4. A decision support system (DSS) based on the EURECA algorithm was implemented. This is a major contribution because DSS that are based on ANN
rule extraction algorithm is rare. The DSS supports the users in terms of: (i) generating multiple linear regression rules for a studied problem using the EURECA algorithm, (ii) providing some insight to the user on the relationships between the predictor and predicted attributes of the studied problem, (iii) performing back calculation, which requires determining the values of some unknown predictor attributes so that the target predicted values could be achieved, and (iv) generating accurate predictions given a new set of predictor attributes of the studied problem.

5. Most of the studies on ANN rule extraction algorithms found evaluated their algorithms by applying them onto UCI datasets, whereas the case studies in this research showed that the EURECA algorithm is suitable for solving industrial engineering problems.

Some detailed conclusions that can be drawn for each of the proposed ANN rule extraction algorithm, the decision support system, and each of the three case studies are presented as follows:

1. The preliminary PWL-ANN algorithm is a decompositional ANN rule extraction algorithm, which extracts rules in the form of multiple linear regression equations from a trained ANN model by approximating each hidden neuron activation with a 3-piece linear equation. The results showed that the best performance was given when the full range of data values was included and when a finer grained approximation was used. The algorithm gave satisfactory fidelity on sixteen out of the nineteen tested datasets, with less than 20% MSE difference between the
PWL-ANN and ANN models. An analysis of the values of $R^2$ given by the PWL approximation on the hidden neurons and the overall output showed that improving the precision of the approximation of each node can improve the predictive accuracy and fidelity of the PWL-ANN models.

2. The enhanced-PWL-ANN algorithm was developed based on the results and analysis of the preliminary PWL-ANN algorithm. The results showed that
   a. By increasing the number of the breakpoints in the selected hidden nodes of the original ANN model, the fidelity of the enhanced-PWL-ANN models derived from the datasets with previously unsatisfactory results from the PWL-ANN models were all improved. The new models developed from application of the enhanced-PWL-ANN algorithm had MSE differences that were less than 20% compared to the originally trained ANN models.
   b. There was a clear trade-off between the fidelity of the enhanced-PWL-ANN models and the comprehensibility of the generated rule sets. Depending on the characteristics of the dataset, the user would need to consider and determine if increasing the breakpoints of select hidden nodes is justified.
   c. The detailed example using Data #6 discussed in Section 4.3 showed that the extracted rule set could be further refined by employing the decision tree algorithm. However, this improvement in comprehensibility could also compromise the accuracy of the generated rule set. This result showed that there was a trade-off between the accuracy and the comprehensibility of the rules.
d. The major drawback of the enhanced-PWL-ANN algorithm was that the rules were generated in terms of the values with which the originally trained ANN model was trained. If the ANN model was trained with normalized values, then the rules would be expressed in terms of normalized values and may not have physical meanings to the domain experts.

3. The EURECA algorithm was developed to overcome the drawback of the enhanced-PWL-ANN algorithm, and the rules given by the EURECA algorithm were in terms of the real values of the dataset. The algorithm was proposed based on the hypothesis that the ANN algorithm assigned the same range of weights to the data tuples with shared properties, and that one of the properties shared by the data tuples within the subset was a linear relationship. The results showed that the models generated by the EURECA algorithm achieved higher predictive accuracy than both the ANN models and the enhanced-PWL-ANN models.

4. A decision support system was developed based on the EURECA algorithm. The EURECA DSS uses the EURECA algorithm to extract rules from trained ANN models and the rules are in the form of multiple linear regression equations. The DSS allows the user to examine the accuracy of the generated EURECA model through the value of MSE and $R^2$ and the user can adjust the number and locations of the breakpoints until satisfactory accuracy of the EURECA model is achieved. The significance of the generated model can be examined through the ANOVA table presented by the system. The DSS provides some insight to the user on the relationship between the predictor and predicted attributes as shown in the coefficients of the attributes in the extracted rules. The DSS also solves the
back calculation problems for the user and suggests the likely value(s) of the unknown predictor attribute(s) that will achieve the target predicted value. Last but not least, the DSS supports predictions using the generated EURECA model.

5. The application of the EURECA algorithm in case study #1 served two purposes. First, the case study illustrated how the extracted rules successfully provided explicit relationships between the predictor and predicted attributes. The EURECA algorithm was able to produce a total of 14 equations with satisfactory predictive accuracy for the water-oil pipeline dataset. By examining the coefficients of the predictor attributes in the generated equations, it can be seen that oil superficial velocity is the most significant parameter that influences the pressure gradient in a two-phase flow pipeline. Secondly, the application case study demonstrated how the extracted rules can provide explicit knowledge by opening the “black-box” of the ANN algorithm. Therefore it enhanced our understanding of both the learning process of the ANN algorithm and the studied problem. By examining the conditions of the rules, it was revealed that the predictor attribute of pipe diameter was the most significant attribute and that there is a correlation between the assigned weight values. The observation suggested a linear relationship exists among the subset of data that shares similar characteristics in their predictor attributes.

6. The EURECA algorithm was applied to the CO₂ production rate dataset obtained from the CO₂ capture process system located at CETRI at the University of Regina. This case study demonstrated the application of the generated EURECA model on solving the back calculation problems that arise during the system
operations. Two levels of user expertise were assumed, and a novice and an expert approach were proposed to solve the back calculation problem using these extracted rules or equations. The novice approach solved the unknown values of the predictor attributes via a brute-force search approach with minimum input from the operator. The solution that gives the lowest average absolute deviation between the calculated and target predicted attribute was selected as the solution. The expert approach was designed for the domain expert, who may wish to have control over the problem-solving process through selection of equations used, estimation of the value of some of the unknowns, and the desired error rate. The results showed that the novice approach was capable of solving up to five unknown predictor attributes values with 20 increments, whereas the expert approach was able to solve two unknown values only. The solutions given by the novice approach had an error or average absolute deviation (AAD) of 0.2535% between the calculated and target predicted attribute values and an AAD of 11.39% between the solved and actual predictor attribute values. In comparison, the solutions given by the expert approach was slightly less accurate with an AAD of 2.53% between the calculated and target predicted attribute values and 14.12% between the solved and actual predictor attribute values. Although the novice approach demonstrated higher accuracy, the expert approach has the major advantages of supporting (i) better control on the problem-solving process, and (ii) user defined granularity of error. Comparing the EURECA approach to previously applied neuro-fuzzy (ANFIS) approach in (Zhou et al., 2013), the EURECA approach could use the same model to solve different operational
problems involving the same set of attributes, whereas the ANFIS approach requires a new model to be trained for each problem scenario. In addition, the selection of predictor parameters was critical when training the new ANFIS models, and if they were not correctly selected, the ANFIS approach would not be able to generate a model. This problem was completely avoided with the EURECA approach.

7. The EURECA algorithm was applied to solve the amine solvent monitoring problem in the CO₂ capture process system. The dataset examined was obtained through experimental work (Pouryousefi, 2014), and the amine solvent was a quaternary system that contained carbon dioxide, water, methyldiethanolamine (MDEA) and monoethanolamine (MEA). The physical properties of the amine solvent studied include density, viscosity, heat capacity, refractive index, thermal conductivity and thermal diffusivity. The results showed that by examining the magnitude of the attribute coefficients in the extracted rules, the concentration of MDEA was identified as the most significant attribute affecting the density of the solvent mixture. The result was validated by the domain expert, and was compared to the relationships between density and the concentrations of each component solvent given by the Weiland correlation. The extracted rules was also used to determine the concentration of each solvent component given the value of the physical property and the temperature at which the property was measured at. The results showed that almost 70% of the solved values had an average absolute deviation of less than 30% when four unknown attributes was presented. The domain expert validated that the rules generated from EURECA could provide a
reasonable guideline when determining the concentration of amine solvent. Compared to the results related to the ANN models presented in (Pouryousefi, 2014), the generated EURECA models were able to determine the concentration of each solvent component more accurately than the ANN models.

10.2 Future Work

Some directions of future work include:

1. Extend the EURECA algorithm such that it is able to extract rules from ANN models with more than one hidden layer.

2. Further extend the functionality of the decision support system by including modules that support visualization, such as 2D- and 3D-plots of the studied dataset.

3. Extend the expert approach so that more than two unknown predictor attributes can be solved.

4. Extend the application of the EURECA models on solving the back calculation problems by developing modules the DSS that supports combining the generated EURECA models, such that the DSS can solve operational problems in the CO$_2$ capture process plant that involve more than one of the four key output parameters of CO$_2$ production rate, heat duty, lean loading, and CO$_2$ absorption efficiency.
REFERENCES


*Thesis. The Norwegian University of Science and Technology.*


Ghiasi, M. M., Arabloo, M., Mohammadi, A. H., & Barghi, T. (2016). Application of ANFIS soft computing technique in modeling the CO2 capture with MEA, DEA, and


Hamzehie, M. E., Mazinani, S., Davardoost, F., Mokhtare, A., Najibi, H., Bruggen, B.
Van Der, & Darvishmanesh, S. (2014). Developing a feed forward multilayer neural
network model for prediction of CO2 solubility in blended aqueous amine solutions.
http://doi.org/10.1016/j.jngse.2014.07.022

Hayashi, Y. (2016). Application of a rule extraction algorithm family based on the Re-
RX algorithm to financial credit risk assessment from a Pareto optimal perspective.
*Operations Research Perspectives, 3*, 32–42.

Hayashi, Y., Setiono, R., & Azcarraga, A. (2016). Neural network training and rule
extraction with augmented discretized input. *Neurocomputing, 207*, 610–622.

Herm Stapelberg, H. & Hewes, D. (1994). The pressure loss and slug frequenoe of liquid-

perceptrons in classification problems: A clustering-based approach.

Huang, J., Gao, J., & Zhang, Y. (2015). Combination of artificial neural network and
clustering techniques for predicting phytoplankton biomass of Lake Poyang, China.

Regression Rule extraction, 270–279.
http://doi.org/http://dx.doi.org/10.1007/11823728_26

Huysmans, J., Baesens, B., & Vanthienen, J. (2006). Using rule extraction to improve the
comprehensibility of predictive models. *Available at SSRN 961358.*


in Cancer Diagnosis. *Journal of Bionanoscience*, 7(6), 665–672.  
http://doi.org/10.1166/jbns.2013.1160

http://doi.org/10.17230/ingciencia.11.22.10

http://doi.org/10.1016/j.eswa.2007.11.024

http://doi.org/10.5121/ijaia.2015.6505

http://doi.org/10.1109/FSKD.2011.6019582

http://doi.org/http://dx.doi.org/10.1007/s10489-006-6929-9


http://doi.org/10.14257/ijseia.2013.7.6.31


http://doi.org/10.1016/S0167-8655(98)00145-7


Tickle, A. B., Andrews, R., Golea, M., & Diederich, J. (1998). The truth will come to light: Directions and challenges in extracting the knowledge embedded within trained


http://doi.org/10.1109/ICNC.2011.6021906


[http://doi.org/10.3390/su8050433](http://doi.org/10.3390/su8050433)
## Appendix A – Rule Set A

<table>
<thead>
<tr>
<th>Rules</th>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
<th>X5</th>
<th>X6</th>
<th>constant</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.00121</td>
<td>-0.00725</td>
<td>0.004772</td>
<td>-0.00422</td>
<td>-0.00603</td>
<td>0.093423</td>
<td>-0.88651</td>
</tr>
<tr>
<td>2</td>
<td>-0.00121</td>
<td>-0.00725</td>
<td>0.004785</td>
<td>-0.00424</td>
<td>-0.00603</td>
<td>0.093452</td>
<td>-0.8865</td>
</tr>
<tr>
<td>3</td>
<td>0.00046</td>
<td>0.000611</td>
<td>0.005239</td>
<td>0.001019</td>
<td>-0.0045</td>
<td>0.080595</td>
<td>-0.89103</td>
</tr>
<tr>
<td>4</td>
<td>0.008827</td>
<td>0.044704</td>
<td>0.016244</td>
<td>0.040741</td>
<td>-0.00495</td>
<td>0.189133</td>
<td>-0.88348</td>
</tr>
<tr>
<td>5</td>
<td>0.018245</td>
<td>0.02515</td>
<td>0.051461</td>
<td>-0.03563</td>
<td>-0.04928</td>
<td>0.688931</td>
<td>-0.97449</td>
</tr>
<tr>
<td>6</td>
<td>0.041492</td>
<td>-0.02311</td>
<td>0.138386</td>
<td>-0.22414</td>
<td>-0.15869</td>
<td>1.922555</td>
<td>-1.65263</td>
</tr>
<tr>
<td>7</td>
<td>0.083193</td>
<td>-0.00494</td>
<td>0.259747</td>
<td>-0.37896</td>
<td>-0.28862</td>
<td>3.563721</td>
<td>-2.97065</td>
</tr>
<tr>
<td>8</td>
<td>-0.00051</td>
<td>-0.00393</td>
<td>0.002761</td>
<td>-0.00028</td>
<td>-0.00634</td>
<td>0.083829</td>
<td>-0.8841</td>
</tr>
<tr>
<td>9</td>
<td>0.068934</td>
<td>-0.08008</td>
<td>0.240993</td>
<td>-0.44665</td>
<td>-0.28784</td>
<td>3.378745</td>
<td>-2.81344</td>
</tr>
<tr>
<td>10</td>
<td>0.007159</td>
<td>0.036839</td>
<td>0.015777</td>
<td>0.0355</td>
<td>-0.00648</td>
<td>0.201961</td>
<td>-0.87896</td>
</tr>
<tr>
<td>11</td>
<td>0.008829</td>
<td>0.0447</td>
<td>0.016231</td>
<td>0.040758</td>
<td>-0.00495</td>
<td>0.189104</td>
<td>-0.8835</td>
</tr>
<tr>
<td>12</td>
<td>0.032504</td>
<td>0.100297</td>
<td>0.070215</td>
<td>0.032064</td>
<td>-0.05005</td>
<td>0.873907</td>
<td>-1.13171</td>
</tr>
<tr>
<td>13</td>
<td>0.055751</td>
<td>0.052033</td>
<td>0.15714</td>
<td>-0.15644</td>
<td>-0.15946</td>
<td>2.107531</td>
<td>-1.80985</td>
</tr>
<tr>
<td>14</td>
<td>0.007856</td>
<td>0.040167</td>
<td>0.013766</td>
<td>0.036907</td>
<td>-0.00679</td>
<td>0.196867</td>
<td>-0.88086</td>
</tr>
<tr>
<td>15</td>
<td>-0.00024</td>
<td>-0.00272</td>
<td>0.00725</td>
<td>-0.00039</td>
<td>-0.00419</td>
<td>0.085688</td>
<td>-0.88913</td>
</tr>
<tr>
<td>16</td>
<td>0.008127</td>
<td>0.041381</td>
<td>0.018268</td>
<td>0.039316</td>
<td>-0.00464</td>
<td>0.194255</td>
<td>-0.88156</td>
</tr>
<tr>
<td>17</td>
<td>-0.00024</td>
<td>-0.00271</td>
<td>0.007261</td>
<td>-0.00039</td>
<td>-0.00418</td>
<td>0.085717</td>
<td>-0.88911</td>
</tr>
<tr>
<td>18</td>
<td>0.009879</td>
<td>-0.01894</td>
<td>0.040456</td>
<td>-0.07535</td>
<td>-0.04882</td>
<td>0.580393</td>
<td>-0.98204</td>
</tr>
<tr>
<td>19</td>
<td>0.017274</td>
<td>0.020613</td>
<td>0.048983</td>
<td>-0.03947</td>
<td>-0.05112</td>
<td>0.696665</td>
<td>-0.97187</td>
</tr>
<tr>
<td>20</td>
<td>0.007156</td>
<td>0.036844</td>
<td>0.01579</td>
<td>0.035482</td>
<td>-0.00648</td>
<td>0.20199</td>
<td>-0.87894</td>
</tr>
<tr>
<td>21</td>
<td>-0.00051</td>
<td>-0.00393</td>
<td>0.002748</td>
<td>-0.0028</td>
<td>-0.00634</td>
<td>0.0883</td>
<td>-0.88843</td>
</tr>
<tr>
<td>22</td>
<td>0.000463</td>
<td>0.000606</td>
<td>0.005226</td>
<td>0.001036</td>
<td>-0.0045</td>
<td>0.080566</td>
<td>-0.89105</td>
</tr>
<tr>
<td>23</td>
<td>0.033126</td>
<td>-0.06721</td>
<td>0.127381</td>
<td>-0.26386</td>
<td>-0.15824</td>
<td>1.814017</td>
<td>-1.66018</td>
</tr>
<tr>
<td>24</td>
<td>0.060567</td>
<td>-0.12418</td>
<td>0.229988</td>
<td>-0.48637</td>
<td>-0.28739</td>
<td>3.270207</td>
<td>-2.82099</td>
</tr>
<tr>
<td>25</td>
<td>0.016574</td>
<td>0.01729</td>
<td>0.051007</td>
<td>-0.04089</td>
<td>-0.05081</td>
<td>0.701788</td>
<td>-0.96995</td>
</tr>
<tr>
<td>26</td>
<td>0.030833</td>
<td>0.092436</td>
<td>0.069762</td>
<td>0.026806</td>
<td>-0.05158</td>
<td>0.886764</td>
<td>-1.12717</td>
</tr>
<tr>
<td>27</td>
<td>0.031533</td>
<td>0.09576</td>
<td>0.067737</td>
<td>0.02823</td>
<td>-0.05189</td>
<td>0.881641</td>
<td>-1.12909</td>
</tr>
<tr>
<td>28</td>
<td>0.05478</td>
<td>0.047496</td>
<td>0.154662</td>
<td>-0.16028</td>
<td>-0.1613</td>
<td>2.115266</td>
<td>-1.80723</td>
</tr>
<tr>
<td>29</td>
<td>0.00813</td>
<td>0.041376</td>
<td>0.018255</td>
<td>0.039334</td>
<td>-0.00464</td>
<td>0.194226</td>
<td>-0.88158</td>
</tr>
<tr>
<td>30</td>
<td>0.017545</td>
<td>0.021827</td>
<td>0.053485</td>
<td>-0.03706</td>
<td>-0.04896</td>
<td>0.694053</td>
<td>-0.97257</td>
</tr>
<tr>
<td>31</td>
<td>0.031804</td>
<td>0.096973</td>
<td>0.072239</td>
<td>0.03064</td>
<td>-0.04974</td>
<td>0.879029</td>
<td>-1.12979</td>
</tr>
<tr>
<td>32</td>
<td>0.055051</td>
<td>0.04871</td>
<td>0.159164</td>
<td>-0.15787</td>
<td>-0.15915</td>
<td>2.112654</td>
<td>-1.80793</td>
</tr>
</tbody>
</table>

*Figure A1 – Rules extracted from dataset #6 (Rule Set A)*
Figure A2 – Range of values (Normalized) covered by each attribute for the Rule Set A

320

Rules

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32

No.of.dataY1Min
30 -0.98799
23 -0.98759
38 -0.94729
10 -0.89459
29 -0.82612
25 -0.58626
23 0.03063
19 -0.97396
20 -0.16299
4 -0.92433
1 -0.82123
4 -0.57844
7 -0.34623
10 -0.92012
9 -0.98165
2 -0.87239
3 -0.9421
11 -0.88119
2 -0.78785
13 -0.93819
1 -0.94572
11 -0.97208
1 -0.4649
1 -0.14948
1 -0.77028
1 -0.64852
2 -0.66581
1 -0.27831
2 -0.89334
2 -0.79807
1 -0.57583
1 -0.31087

Y1Max
X1Min
X1Max
X2Min
X2Max
X3Min
X3Max
X4Min
X4Max
X5Min
X5Max
X6Min
-0.93502
-1
1
-1
1
-1
1 -0.25984
1
-1
0.2967
-1
-0.92218
-1
1 -0.02857
1
0.075
0.9 -0.8189 0.12598 -0.07692
1
-1
-0.88341
-1
1
-1 0.11429
-1
0.925
-1 0.03937 -0.25275
1 -0.69231
-0.79214
-1
1 -0.02857
1
0.075
0.9 -0.18898
1 -0.93407 0.75824 -0.07692
-0.59117
-1
1
-1
1
-1
1 -0.86614
1 -0.93407
1 0.23077
-0.20896
-1
1
-1 0.25714
-1
0.925 -0.86614 0.03937 -0.93407
1 0.53846
0.74565
-1
1 -0.08571
1
-1
1 -0.18898
1
-1 0.75824 0.84615
-0.89643
-1
1
-1 0.11429
0.075
1 -0.8189 0.68504 -0.07692
1
-1
0.87871
-1
1
-1 0.25714
-1
0.925
-1 0.03937 -0.93407
1 0.69231
-0.85079
0.08
0.08
0
1
-1
0.1 0.11811
1
-1 -0.93407 -0.38462
-0.82123
0.08
0.08
0
0
0.1
0.1
1
1 -0.93407 -0.93407 0.07692
-0.51343
-1
1 -0.08571
1
0.1
1 0.12598
1 -0.93407 -0.03297 0.53846
-0.01387
-1
1 -0.08571
1
0.1
1 -0.18898
1 -0.93407
0.2967 0.69231
-0.81477
0.04
1 -0.08571
1
0.1
1 -0.18898 0.68504 -0.03297 0.75824 -0.38462
-0.89937
0.04
0.08
-1 0.25714
-1
-0.95
-1 -0.09449 -0.93407 -0.07692
-1
-0.83358
0.04
0.08 0.25714
1
-1
-0.95 -0.09449 0.11811
-1 -0.93407 0.07692
-0.91573
0.04
0.04 0.08571 0.08571
-1
-1 -0.86614 -0.86614 -0.07692 -0.07692 -0.53846
-0.70043
-1
1
-1 0.08571
-1
0.925
-1 -0.25984 -0.07692 0.71429 0.07692
-0.78433
0.08
0.08 -0.08571
1
0.9
1 0.07087 0.68504 -0.03297 0.71429 0.23077
-0.81286
-1
1 0.57143
1
0.1
0.9 -0.18898 0.12598 -0.07692 0.71429 -0.53846
-0.94572
0.12
0.12 -0.54286 -0.54286
0.1
0.1 0.03937 0.03937 -0.25275 -0.25275 -0.69231
-0.933
-1
1
-1 -0.54286
-1
0.1
-1 0.03937 -0.25275 -0.05495
-1
-0.4649
0.08
0.08
-1
-1
-1
-1
-1
-1 -0.07692 -0.07692 0.53846
-0.14948
0.08
0.08
-1
-1
-1
-1
-1
-1 -0.07692 -0.07692 0.69231
-0.77028
1
1
1
1
0.1
0.1 0.12598 0.12598 -0.07692 -0.07692 0.23077
-0.64852
1
1
1
1
0.1
0.1 0.12598 0.12598 -0.07692 -0.07692 0.38462
-0.53017
0.08
0.08
1
1
0.9
0.9 0.07087 0.07087 0.71429 0.71429 0.38462
-0.27831
0.08
0.08
1
1
0.9
0.9 0.07087 0.07087 0.71429 0.71429 0.69231
-0.86346
0.08
0.08
1
1
-1
-1 0.11811 0.11811
-1
-1 -0.23077
-0.6913
0.08
0.08
1
1
-1
-1 0.11811 0.11811
-1
-1 0.23077
-0.57583
0.08
0.08
1
1
-1
-1 0.11811 0.11811
-1
-1 0.53846
-0.31087
0.08
0.08
1
1
-1
-1 0.11811 0.11811
-1
-1 0.69231

X6Max
-0.53846
-0.38462
0.07692
0.23077
0.53846
0.69231
1
-0.07692
1
-0.07692
0.07692
0.53846
0.84615
0.07692
-0.07692
0.07692
-0.23077
0.38462
0.23077
0.07692
-0.69231
-0.53846
0.53846
0.69231
0.23077
0.38462
0.53846
0.69231
-0.07692
0.38462
0.53846
0.69231


Appendix B – Rule Set B

1. IF \((x_6 \geq 0.77) \text{ AND } (x_4 \geq -0.22)\) THEN \(y_1 = \text{Equation 7}\)

2. IF \((x_6 \geq 0.77) \text{ AND } (x_4 < -0.22)\) THEN \(y_1 = \text{Equation 9}\)

3. IF \((x_6 < -0.62) \text{ AND } (x_5 \geq 0.13)\) THEN \(y_1 = \text{Equation 2}\)

4. IF \((x_6 < -0.62) \text{ AND } (x_5 < 0.13) \text{ AND } (x_4 \geq -0.18)\) THEN \(y_1 = \text{Equation 1}\)

5. IF \((x_6 < -0.62) \text{ AND } (x_5 < 0.13) \text{ AND } (x_4 < -0.18)\) THEN \(y_1 = \text{Equation 22}\)

6. IF \((0.15 > x_6 \geq -0.62) \text{ AND } (x_2 < -0.31)\) THEN \(y_1 = \text{Equation 3}\)

7. IF \((0.15 > x_6 \geq -0.31) \text{ AND } (0.79 > x_2 \geq -0.31) \text{ AND } (x_3 < 0.5) \text{ AND } (x_5 \geq -0.51)\) THEN \(y_1 = \text{Equation 3}\)

8. IF \((-0.31 > x_6 \geq -0.62) \text{ AND } (0.79 > x_2 \geq -0.31) \text{ AND } (x_3 < 0.5) \text{ AND } (x_5 \geq -0.51)\) THEN \(y_1 = \text{Equation 2}\)

9. IF \((0.15 > x_6 \geq -0.62) \text{ AND } (0.79 > x_2 \geq -0.31) \text{ AND } (x_3 < 0.5) \text{ AND } (x_5 < -0.51)\) THEN \(y_1 = \text{Equation 15}\)

10. IF \((-0.15 > x_6 \geq -0.62) \text{ AND } (0.79 > x_2 \geq -0.31) \text{ AND } (x_3 \geq 0.5)\) THEN \(y_1 = \text{Equation 8}\)

11. IF \((0.77 > x_6 \geq -0.15) \text{ AND } (0.79 > x_2 \geq -0.31) \text{ AND } (x_3 \geq 0.5)\) THEN \(y_1 = \text{Equation 14}\)

12. IF \((0.77 > x_6 \geq 0.15) \text{ AND } (x_2 \geq 0.79)\) THEN \(y_1 = \text{Equation 20}\)

13. IF \((0.77 > x_6 \geq 0.46) \text{ AND } (x_4 < 0.055)\) THEN \(y_1 = \text{Equation 6}\)

14. IF \((0.77 > x_6 \geq 0.46) \text{ AND } (x_4 \geq 0.055)\) THEN \(y_1 = \text{Equation 12}\)

15. IF \((0.46 > x_6 \geq 0.15) \text{ AND } (x_2 \geq -0.77)\) THEN \(y_1 = \text{Equation 5}\)

16. IF \((0.46 > x_6 \geq 0.15) \text{ AND } (x_2 < -0.77)\) THEN \(y_1 = \text{Equation 18}\)
<table>
<thead>
<tr>
<th>Equations</th>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
<th>X5</th>
<th>X6</th>
<th>constant</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.00121</td>
<td>-0.00725</td>
<td>0.004772</td>
<td>-0.00422</td>
<td>-0.00603</td>
<td>0.093423</td>
<td>-0.88651</td>
</tr>
<tr>
<td>2</td>
<td>-0.00121</td>
<td>-0.00725</td>
<td>0.004785</td>
<td>-0.00424</td>
<td>-0.00603</td>
<td>0.093452</td>
<td>-0.8865</td>
</tr>
<tr>
<td>3</td>
<td>0.00046</td>
<td>0.000611</td>
<td>0.005239</td>
<td>0.001019</td>
<td>-0.0045</td>
<td>0.080595</td>
<td>-0.89103</td>
</tr>
<tr>
<td>4</td>
<td>0.018245</td>
<td>0.02515</td>
<td>0.051461</td>
<td>-0.03563</td>
<td>-0.04928</td>
<td>0.688931</td>
<td>-0.97449</td>
</tr>
<tr>
<td>5</td>
<td>0.041492</td>
<td>-0.02311</td>
<td>0.138386</td>
<td>-0.22414</td>
<td>-0.15869</td>
<td>1.922555</td>
<td>-1.65263</td>
</tr>
<tr>
<td>6</td>
<td>0.083193</td>
<td>-0.00494</td>
<td>0.259747</td>
<td>-0.37896</td>
<td>-0.28862</td>
<td>3.563721</td>
<td>-2.97065</td>
</tr>
<tr>
<td>7</td>
<td>-0.00051</td>
<td>-0.00393</td>
<td>0.002761</td>
<td>-0.00282</td>
<td>-0.00634</td>
<td>0.088329</td>
<td>-0.88841</td>
</tr>
<tr>
<td>8</td>
<td>0.068934</td>
<td>-0.08008</td>
<td>0.240993</td>
<td>-0.44665</td>
<td>-0.28784</td>
<td>3.378745</td>
<td>-2.81344</td>
</tr>
<tr>
<td>9</td>
<td>0.032504</td>
<td>0.100297</td>
<td>0.070215</td>
<td>0.032064</td>
<td>-0.05005</td>
<td>0.873907</td>
<td>-1.13171</td>
</tr>
<tr>
<td>10</td>
<td>0.007856</td>
<td>0.040167</td>
<td>0.013766</td>
<td>0.036907</td>
<td>-0.00679</td>
<td>0.196867</td>
<td>-0.88086</td>
</tr>
<tr>
<td>11</td>
<td>-0.00024</td>
<td>-0.00272</td>
<td>0.00725</td>
<td>-0.00039</td>
<td>-0.00419</td>
<td>0.085688</td>
<td>-0.88913</td>
</tr>
<tr>
<td>12</td>
<td>0.009879</td>
<td>-0.01894</td>
<td>0.040456</td>
<td>-0.07535</td>
<td>-0.04882</td>
<td>0.580393</td>
<td>-0.98204</td>
</tr>
<tr>
<td>13</td>
<td>0.007156</td>
<td>0.036844</td>
<td>0.01579</td>
<td>0.035482</td>
<td>-0.00648</td>
<td>0.20199</td>
<td>-0.87894</td>
</tr>
<tr>
<td>14</td>
<td>0.000463</td>
<td>0.000606</td>
<td>0.005226</td>
<td>0.001036</td>
<td>-0.0045</td>
<td>0.080566</td>
<td>-0.89105</td>
</tr>
</tbody>
</table>

*Figure B1 – Rule Set B Equations*
Appendix C – Justification and Performance of the Sampling Algorithm

After extending the brute-force search of the PWL-ANN algorithm for searching more than one breakpoint (BP), the search algorithm (or approximation algorithm, as the algorithm approximates the function of one hidden neuron into a piece-wise linear or PWL function) was tested on datasets of different sizes, and it was found that it can be very time consuming to approximate datasets with greater than 1000 points for more than 1 breakpoint. From the evaluation in Section C1, it can be seen that the computation time required was in an exponential relationship with the data size as well as the number of desired breakpoints. Therefore, in order to improve the time efficiency of the search algorithm, a sampling algorithm is included. The sampling algorithm generates a subset of data by taking every x point in the data set and x is given by actual data size divided by the desired data size of subset. Before deciding the sampling algorithm was part of the enhanced-PWL-ANN algorithm, the performance of the sampling algorithm in terms of accuracy in the PWL approximation was tested and results can be found in Section C.2. Based on the results, it can be seen that the sampling algorithm give compatible accuracies to using the full dataset with considerably less time.

C.1 Data Size and Number of Breakpoints Vs Computation Time

Table C1 shows the computation time required to find 1 breakpoint (BP) and 2 BPs in seconds for different data size. The results are also plotted in Figure C1 and Figure C2 for computation time verses 1 BP and 2 BPs, respectively.
Table C1 – Data Size and number of breakpoints Vs computation time

<table>
<thead>
<tr>
<th>Data Size</th>
<th>BP = 1, Time (sec)</th>
<th>BP = 2, Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>0.6</td>
<td>45.7</td>
</tr>
<tr>
<td>600</td>
<td>1.4</td>
<td>194.7</td>
</tr>
<tr>
<td>1000</td>
<td>2.7</td>
<td>560.0</td>
</tr>
<tr>
<td>2000</td>
<td>6.8</td>
<td>2,696.3</td>
</tr>
<tr>
<td>10000</td>
<td>94.2</td>
<td>176,717.8</td>
</tr>
</tbody>
</table>
Figure C1 – Computation time Vs data size for 1 breakpoint

Figure C2 – Computation time Vs data size for 2 breakpoints
From the results, it can be seen that the time required to estimate 2 BPs increased significantly compared to the time required to estimate one BP. For a dataset with 300 data tuples, the approximation required 0.6 seconds to estimate one BP, and 46 seconds for two breakpoints. When a dataset of 10,000 data tuples was given, the computation time required to estimate 2 breakpoints was 176,718 seconds, which was approximately 49 hours, compared to 94 seconds for 1 breakpoint. This can be problematic because the search or approximation algorithm needs to be repeated for the number of hidden nodes that is present in the trained ANN model. For example, if the trained ANN model for the dataset with 10,000 data tuples has 6 hidden nodes, the estimated computation time to approximate each of the hidden nodes by a PWL function with 2 breakpoints is 176,718, and for the six hidden nodes is at least 176,718 seconds multiplied by 6, which is approximately two weeks. Therefore, to reduce the computation time required by the approximation algorithm, and to improve the overall computation efficiency of the enhanced-PWL-ANN algorithm, a sampling algorithm is included as an option for the users. The sampling algorithm generates a subset of data by taking every x point in the data set and x is given by actual data size divided by desired data size of subset and the performance of the sampling algorithm is evaluated in the next section.

C.2 Effect of Sampling on Performance of the Approximation Algorithm

The effects of the sampling algorithm on the performance of the approximation algorithm were tested on the hidden nodes of 4 datasets: dataset #1, #10, #12 and #17. These datasets were chosen because they covered a range of data size of 1,000 to above 10,000. The absolute values of weighted input of each hidden node of the datasets were
approximated into piece-wise linear (PWL) equations by the enhanced brute-force search or approximation algorithm to two breakpoints using

i) full dataset;

ii) subset size of 1000, and;

iii) subset size of 300.

The performances in terms of accuracies were evaluated using the value of $R^2$, and the performances in terms of computation efficiency were evaluated based on the number of seconds required to find the best PWL equations for one hidden node. The characteristics of the datasets and the results of the PWL approximations are summarized in Table C2 and Table C3, respectively. Table C.3 shows the average time required, and the average $R^2$ of the approximated PWL of each hidden node. For the subsets, the change of $R^2$, $\Delta R^2$, from the $R^2$ of the full data size, i.e.

$$\Delta R^2 = R^2_{\text{full}} - R^2_{\text{subset}},$$

were calculated in order to compare the accuracy of the approximation given when using the full dataset and the subsets.

As shown in Table C3, the accuracies of the PWL equations were not affected when data size was decreased from full size to 1000 points for datasets #10 and #12. The $\Delta R^2$ in these cases were zeros. There was a slight decrease in $R^2$ for dataset #1 and #10 when the sample size was decreased to 300, and the values of $\Delta R^2$ were -0.0004 and -0.0002, respectively. On the other hand, the accuracies of the PWL equations for Dataset #17 were affected more significantly by using the sampling algorithm. The $R^2$ values given by the approximation of the subset of size of 1000 and 300 were reduced by 0.0027 and
0.1103, respectively. A decrease of 0.1103 in $R^2$ may not be ideal, however, it could be justified by the reduction in time: from 269051 seconds (74.7 hours or just over 3 days) to 54.4 seconds (less than a minute).
### Table C2 – Characteristics of datasets

<table>
<thead>
<tr>
<th>#</th>
<th>Dataset</th>
<th>Data size</th>
<th>No. of Input attribute</th>
<th>No. of input nodes</th>
<th>No. of hidden nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>Energy Efficiency Heat loading (UCI)</td>
<td>767</td>
<td>8</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>#10</td>
<td>Parkinson (UCI)</td>
<td>5873</td>
<td>19</td>
<td>19</td>
<td>12</td>
</tr>
<tr>
<td>#12</td>
<td>ERA (Weka)</td>
<td>1000</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>#17</td>
<td>Heavy Duty</td>
<td>10421</td>
<td>8</td>
<td>8</td>
<td>6</td>
</tr>
</tbody>
</table>

### Table C3 – Performance of approximation algorithm using different data size

<table>
<thead>
<tr>
<th></th>
<th>Full Data Size</th>
<th>Subset Size = 1000</th>
<th>Subset Size = 300</th>
<th>( \Delta R^2 )</th>
<th>( \Delta R^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average Time (s)</td>
<td>Average R^2</td>
<td>Average Time (s)</td>
<td>Average R^2</td>
<td>( \Delta R^2 )</td>
</tr>
<tr>
<td>#1</td>
<td>326.4</td>
<td>0.9991</td>
<td>326.4</td>
<td>0.9991</td>
<td>0.0000</td>
</tr>
<tr>
<td>#10</td>
<td>42308.3</td>
<td>0.9978</td>
<td>809.0</td>
<td>0.9978</td>
<td>0.0000</td>
</tr>
<tr>
<td>#12</td>
<td>580.4</td>
<td>0.9960</td>
<td>580.4</td>
<td>0.9960</td>
<td>0.0000</td>
</tr>
<tr>
<td>#17</td>
<td>269051.2</td>
<td>0.9951</td>
<td>723.4</td>
<td>0.9924</td>
<td>-0.0027</td>
</tr>
</tbody>
</table>
The effect of the sampling algorithm on the accuracy and computation efficiency of the approximation algorithm was also analyzed in terms of percentage of subset size to the full data size, and the results are showed in Table C4. The subset size relative to full size was obtained by calculating the ratio between the subset size and the original data size, e.g. 2.88% was obtained from the subset size 300 for data #17, 300*100%/10,421 = 2.88%. Likewise, the time required relative to full data size was obtained by calculating the ratio between the time required for the subset and for the full dataset, e.g. 0.02% was also obtained from the subset size 300 for data #17, 54.4*100%/269051 = 0.02%. It can be observed that by reducing the data size to 2.88% of the original data size, the time required to approximate 2 BPs was reduced to 0.02% of the time that would be required for the original data size. From the results, it can be shown that the accuracy of the approximation was well maintained for subsets size greater than 10% (non-italic values in Table C4). However, there is some trade-off of accuracy once the subset size is decreased to less than 10% of the original size (italic values in Table C4). The user will have to decide if this trade-off of accuracy is justified by the significant reduction in computation time.
Table C4 – Relative size of subsets and relative performance of the approximation algorithm to the full dataset

<table>
<thead>
<tr>
<th>Subset size relative to full data size (%)</th>
<th>$\Delta R^2$</th>
<th>Time Required relative to full data size (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.88</td>
<td>-0.1103</td>
<td>0.02</td>
</tr>
<tr>
<td>5.11</td>
<td>-0.0002</td>
<td>0.11</td>
</tr>
<tr>
<td>9.60</td>
<td>-0.0027</td>
<td>0.27</td>
</tr>
<tr>
<td>17.03</td>
<td>0.0000</td>
<td>1.91</td>
</tr>
<tr>
<td>30.00</td>
<td>0.0000</td>
<td>10.59</td>
</tr>
<tr>
<td>39.16</td>
<td>-0.0004</td>
<td>11.09</td>
</tr>
</tbody>
</table>