AN OPTIMISED
REGION AND BOUNDARY CLASSIFIER
FOR HEAD MOVEMENT CLASSIFICATION

Sean Williams

Doctor of Philosophy in Engineering

2009

CERTIFICATE OF AUTHORSHIP/ORIGINALITY

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Student

_____

ACKNOWLEDGEMENTS

I would like to acknowledge the following people:

- My supervisor Professor Hung Nguyen for his contributions and help throughout this process of research and learning;
- Phillip Taylor and his contribution to the UTS Wheelchair Control Research Group, particularly his efforts in collecting head movement data from volunteers;
- Dr James Middleton for supervising the collection of this data at the Moorong Spinal Unit, Royal Rehabilitation Centre Sydney;
- All those not mentioned who have assisted in some way and helped to make this course of research a challenging yet satisfying experience; and
- Finally, friends and loved ones who put up with my frustrations and obsessions during this period, thankyou.

TABLE OF CONTENTS

Chapter 1 - Introduction 12

1.1 MOTIVATION 12
1.2 ACCURACY & PREDICTABILITY 14
1.3 AIMS OF RESEARCH 17
1.3.1 Contribution 17
1.4 OUTLINE OF THESIS 18

Chapter 2 - Literature Review 20

2.1 PROBLEM STATEMENT 20
2.2 INTRODUCTION 20
2.3 HOW OTHERS ADDRESS THE PROBLEM 23
2.3.1 Discussion 32
2.4 REPRESENTATIVE DATA & EFFECTIVE BOUNDARIES 34
2.4.1 Proposal 35
2.5 DISCUSSION AND CONCLUSION 37

Chapter 3 - Effective Head Movement Boundaries with an ANN 39

3.1 INTRODUCTION 39
3.2 ANN PERFORMANCE 44
3.2.1 Estimated and Actual Performance Discrepancies 45
3.2.1.1 Representative Data 45
3.2.2 Outside the Boundary of the Training Set 48
3.2.3 Trimming the Problem Tree 49
3.2.4 Redefining the Problem Tree 53
3.2.5 Summary and Problem Statement 54
3.3 REPRESENTATIVE DATA & EFFECTIVE BOUNDARIES – DERIVATION 55
3.3.1 Creating Effective Boundaries with Training Data 58
3.3.1.1 Identifying the True Class Boundary 61
3.3.1.2 Forming Effective & Representative Class Boundaries 62
3.3.1.3 Obtaining Effective Training Data 63
3.3.1.3.1 Real Data 63
3.3.1.3.2 Artificial Data 64
3.3.2 Explicit and Implicit Boundaries 65
3.4 DISCUSSION AND CONCLUSION 68

Chapter 4 - Bootstrap and ROC Analysis for Head Movement 70

4.1 INTRODUCTION 70
4.2 TESTING CLASSIFIER ALGORITHMS 70
4.2.1 ROC Analysis 70
4.2.2 The 0.632+ Bootstrap Method 71
4.2.3 The 0.632+ Bootstrap ROC Analysis 72
4.3 CLASSIFIER TYPES UNDER TEST 72
4.3.1 Artificial Neural Networks 73
4.3.2 Hyper-Rectangle Basis Function 74
4.3.3 A Region and Boundary Classifier 81
4.3.4 Definitions 84
4.4 DATA COLLECTION 87
4.4.1 Overview 87

4.4.2 Data Source *87*
4.4.3 System Signal Flow *89*
4.5 PATTERN SET CREATION 91
4.5.1 Manual Viewing and Labelling Tool *94*
4.5.2 Example Patterns *96*
4.5.2.1 Command Signals 96
4.5.2.2 Non-Command Signals 97
4.6 BOOTSTRAP TRAINING & TEST SET CREATION 102
4.7 CLASSIFIER TRAINING 103
4.7.1 Neural Network Training *103*
4.7.2 Hyper-Rectangle Basis Function Training *103*
4.7.3 Region and Boundary Classifier Training *104*
4.8 CLASSIFIER TESTING 106
4.8.1 Accuracy Estimation for Individual Classes *109*
4.9 CLASSIFIER COMPARISONS 109
4.9.1 Classifier Comparison Scoring & Ranking System *109*
4.10 DISCUSSION AND CONCLUSION 110

Chapter 5 - Implicit and Explicit Boundaries for ANN 111

5.1 METHODOLOGY 111
5.1.1 The Questions Posed *114*
5.1.2 The Experiment *115*
5.1.3 Terminology Explanations *115*
5.2 EXPERIMENTAL METHOD 116
5.2.1 Overview *116*
5.2.2 Classifier Identification *117*
5.3 RESULTS 117
5.3.1 ROC Analysis *117*
5.3.1.1 Results Data 117
5.3.1.1.1 NN1 118
5.3.1.1.2 NN2 119
5.3.1.1.3 NN3 120
5.3.1.1.4 HRBF1 121
5.3.1.1.5 All ROC Results 122
5.3.1.2 ROC Comparison & Ranking 123
5.4 DISCUSSION AND CONCLUSION 124

Chapter 6 - An Advanced Boundary Concept 128

6.1 METHODOLOGY 128
6.1.1.1 Improving on the Optimised Hyper-Rectangle Basis Function 130
6.1.2 The Questions Posed *136*
6.1.3 The Experiment *136*
6.1.4 Terminology Explanations *137*
6.2 EXPERIMENTAL METHOD 137
6.2.1 Overview *137*
6.2.2 Classifier Identification *139*
6.3 RESULTS 139
6.3.1 ROC Analysis *139*
6.3.1.1 Results Data 139
6.3.1.1.1 RBC1 140
6.3.1.1.2 All ROC Results 141

6.3.1.2 ROC Comparison & Ranking 142
6.4 DISCUSSION AND CONCLUSION 143

Chapter 7 - Conclusion 146

7.1 SUMMARY 146
7.2 SIGNIFICANCE 151
7.3 FUTURE RESEARCH 153

Bibliography 155

LIST OF FIGURES

Figure 1-1: Hands Free Technology Context 15
Figure 1-2: The Problem 16
Figure 2-1: Generalisation – Single Class 21
Figure 2-2: Problem/Solution Tree 34
Figure 3-1: Desired Generalisation Ideal 40
Figure 3-2: ANN Trained with 4 Classes (2-Dimensional Example) 41
Figure 3-3: Over Generalisation Ideal 42
Figure 3-4: Under Generalisation Ideal 43
Figure 3-5: Problem Tree 44
Figure 3-6: Under Representative Data 46
Figure 3-7: Representative Data 47
Figure 3-8: Problem Tree – First Pass Trim 50
Figure 3-9: Problem Tree – Second Pass Trim 51
Figure 3-10: Final Problem Tree 53
Figure 3-11: Problem/Solution Tree 55
Figure 3-12: Likely Inputs and a Training Set 56
Figure 3-13: Ideal Result of Training 57
Figure 3-14: Likely Result of Training – Over Generalisation 57
Figure 3-15: Likely Result of Training – Under Generalisation 58
Figure 3-16: NN 2D Plot -1 Class, 1 Training Input 59
Figure 3-17: NN 2D Plot -1 Class, 1 Positive & 1 Negative Training Input 60
Figure 3-18: Positive and Negative Patterns and Close to Ideal Generalisation 62
Figure 3-19: The Do not Care Zone or Buffer 62
Figure 3-20: Artificially Generated Data – Shortcomings 64
Figure 3-21: Artificially Generated Data with an Optimised Hyper-Box 65
Figure 3-22: Implicit Boundary Formation 66
Figure 3-23: Explicit Boundary Formation 66
Figure 3-24: Explicit Boundary Formation - Expansion 67
Figure 3-25: Explicit Boundary Formation – Controlled Expansion 67
Figure 3-26: Explicit Boundary Formation – Optimised 68
Figure 4-1: Neural Network Architecture 73
Figure 4-2: Hyper-Rectangle Network Architecture 74
Figure 4-3: Hyper-Rectangle Envelope – Forward (Artificial) 75
Figure 4-4: Hyper-Rectangle Envelope – Forward (Real Data) 75
Figure 4-5: Hyper-Rectangle Envelope – Backward (Artificial) 76
Figure 4-6: Hyper-Rectangle Envelope – Backward (Real Data) 76
Figure 4-7: Hyper-Rectangle Envelope – Left (Artificial) 77
Figure 4-8: Hyper-Rectangle Envelope – Left (Real Data) 77
Figure 4-9: Hyper-Rectangle Envelope – Right (Artificial) 78
Figure 4-10: Hyper-Rectangle Envelope – Right (Real Data) 78
Figure 4-11: Hyper-Rectangle Envelope – All Classes (Real Data) 79
Figure 4-12: Hyper-Rectangle Envelope Multiple Clusters – Forward (Real Data) 80
Figure 4-13: Region and Boundary Classifier Network Architecture 81
Figure 4-14: Region and Boundary Classifier Horizontal Null Region - Artificial 82
Figure 4-15: Region and Boundary Classifier Horizontal Null Region - Real 82
Figure 4-16: Region and Boundary Classifier Horizontal Null Region - Real 83
Figure 4-17: Region and Boundary Classifier Vertical Null Region 84
Figure 4-18: System Signal Flow 89

Figure 4-19: X-axis Accelerometer Real Time Data Signal 90
Figure 4-20: Y-axis Accelerometer Real Time Data Signal 90
Figure 4-21: Manual Labelling Tool – 190 Samples Wide View 94
Figure 4-22: Manual Labelling Tool – 20 Samples Wide View 95
Figure 4-23: Valid Command Training Patterns – R, F, L, B 96
Figure 4-24: Non-Command Training Pattern – Flat Line or Null 97
Figure 4-25: Non-Command Training Patterns – Xp, Yp, Xn, Yn 98
Figure 4-26: Non-Command Training Patterns – XpYp, XpYn, XnYp, XnYn 99
Figure 4-27: Non-Command Training Patterns – FXp, FXn, BXp, BXn 100
Figure 4-28: Non-Command Training Patterns – LYp, LYn, RYp, RYn 101
Figure 5-1: Encapsulation - NN Equivalence with Hyper-Geometric Classifier 113
Figure 5-2: NN1 ROC Curves 118
Figure 5-3: NN2 ROC Curves 119
Figure 5-4: NN3 ROC Curves 120
Figure 5-5: HRBF1 ROC Curves 121
Figure 5-6: Comparison of ROC Curves 122
Figure 6-1: Hyper-Rectangle Boundary 130
Figure 6-2: Hyper-Rectangle Boundary – with Likely Inputs 130
Figure 6-3: Hyper-Rectangle Boundary Expanded (Optimised) to Known Points 131
Figure 6-4: Optimised Hyper-Rectangle Boundary – with Likely Inputs 131
Figure 6-5: Hyper-Rectangle with Additional Regions 132
Figure 6-6: Optimised Hyper-Rectangle Basis Function – All Classes (Real Data) 133
Figure 6-7: Optimised Hyper-Rectangle Basis Function with Null Regions 134
Figure 6-8: Region and Boundary Classifier Network Architecture 135
Figure 6-9: RBC1 ROC Curves 140
Figure 6-10: Comparison of ROC Curves 141

LIST OF TABLES

Table 3-1: Effect of Boundary Control, Knowledge, and Training/Test Data.....52
Table 4-1: Originally Collected and Labelled Head Movement Data - Command 88
Table 4-2: Originally Collected Head Movement Data – Non-Command 88
Table 4-3: Final Number of Patterns Used 93
Table 4-4: Non-Command Pattern Types Used.....93
Table 4-5: Bootstrap Training & Test Set Sizes 102
Table 5-1: Classifier Acronym Definitions.....117
Table 5-2: NN1 ROC Results 118
Table 5-3: NN1 ROC Accuracy Results.....118
Table 5-4: NN1 ROC Range Results 118
Table 5-5: NN2 ROC Results 119
Table 5-6: NN2 ROC Accuracy Results.....119
Table 5-7: NN2 ROC Range Results 119
Table 5-8: NN3 ROC Results 120
Table 5-9: NN3 ROC Accuracy Results.....120
Table 5-10: NN3 ROC Range Results 120
Table 5-11: HRBF1 ROC Results.....121
Table 5-12: HRBF1 ROC Accuracy Results 121
Table 5-13: HRBF1 ROC Range Results.....121
Table 5-14: All ROC Results 122
Table 5-15: All ROC Accuracy Results.....122
Table 5-16: All ROC Range Results.....122
Table 5-17: All ROC Results 123
Table 5-18: ROC Comparison Matrix.....123
Table 5-19: ROC Ranking Matrix.....123
Table 6-1: Classifier Acronym Definitions.....139
Table 6-2: RBC1 ROC Results 140
Table 6-3: RBC1 ROC Accuracy Results.....140
Table 6-4: RBC1 ROC Range Results.....140
Table 6-5: All ROC Results 141
Table 6-6: All ROC Accuracy Results.....141
Table 6-7: All ROC Range Results.....141
Table 6-8: All ROC Results 142
Table 6-9: ROC Comparison Matrix 142
Table 6-10: ROC Ranking Matrix.....142

ABSTRACT

The use of Artificial Neural Network (ANN) classifiers in real-time applications that could be considered critical, such as head movement classification for the control of a powered wheelchair, naturally raises questions over safety due to performance accuracy.

One inherent characteristic of an ANN is that the placement of classification boundaries within the decision space is arbitrary and ultimately unknown. The result of this characteristic is to give unpredictable results when presented with data outside the boundary of the training set and is therefore considered as a major source of uncertainty.

The problem is one of accuracy and predictability and to address this a novel algorithm termed an Optimised Region and Boundary Classifier (RBC) was created to provide improvements in accuracy and predictability over a conventionally trained ANN.

To improve accuracy the RBC requires the formation of effective boundaries, which relies on the training set containing data that is both representative of all types of data likely to be input to the classifier and is complementary (data either side of an implied boundary). To achieve this the original training set consisting of commands only (forward, back, left, and right) was augmented with “data outside the boundary”, which consisted of seventeen types of non-command data that the classifier was likely to see.

To improve predictability the RBC uses explicit boundaries, which is achieved using k-means clustering techniques to define Hyper-Rectangles. Also, two additional regions (vertical and horizontal null regions) are extracted from the Hyper-Rectangles and added to the classifier.

To further improve accuracy and predictability an optimisation process is used that expands the Hyper-Rectangles for each of the classes to be classified (forward, back, left, and right) until the associated training error for sensitivity and specificity is optimal.

To show that the RBC could provide improvements in performance comparisons were made between the RBC trained on the augmented training set and ANN’s trained on both the original training set and the augmented training set. The performance for each type of classifier was assessed using the 0.632+ Bootstrap Method and Receiver

Operating Characteristics (ROC) analysis (area under the curve, sensitivity, and specificity).

Results showed that the RBC provided significant improvements in performance when compared with an ANN trained on the original training set (up to 9% improvement in mean sensitivity and 30% improvement in mean specificity). When compared with the ANN that was trained on the same augmented training data only small improvements in mean sensitivity and specificity could be seen (up to 3%). However, the RBC was clearly the best performing classifier algorithm overall.

The significance of the Optimised Region and Boundary Classifier is that it addresses both accuracy and predictability and therefore it is potentially an inherently safer classification algorithm. This would enable its use with more confidence in applications where safety is critical, such as in Medical Devices.

Chapter 1 - Introduction

1.1 Motivation

ALGORITHMS AND SAFETY

Technologies that use algorithms for pattern recognition and classification, such as an Artificial Neural Network need to be assessed in terms of the safety criticality of the application. An application such as medical imaging and diagnosis for cancer may have profound effects if it incorrectly diagnoses a patient. In this context regulators such as the U.S. Food and Drug Administration (FDA) require that such technologies be thoroughly assessed in terms of the direct and indirect effects and the underlying software and algorithms extensively tested and validated [1].

With this in mind, the use of classifier algorithms in real-time environments for decision-making presents unique problems that are inherent to the use of such algorithms, particularly knowing where decision boundaries are placed. Using classifier algorithms in critical systems may undermine the safety of these systems and present safety risks to users, property, and bystanders alike.

Beyond safety issues the incorrect functioning of the system is of no benefit and from a user point of view a nuisance especially when correct function is critical and money has been spent. It is also of no benefit for a company to sell equipment or devices that cannot be relied upon to function correctly, such companies at the very worst can expect to be out of business very soon.

Taking the example of head movement for wheelchair control [2-4]. An ANN is used to detect and decode head movement into forward, backward, left, and right commands for the control of the wheelchair. If an incorrect classification occurs the user could quite easily have their safety compromised, as control is lost even if for a small amount of time. A user is also likely to panic when control is lost, thereby compounding a problem and when a quick decision is required to regain control this may not happen before safety is compromised.

Safety issues and risk are at the heart of what may be considered a critical system such as wheelchair control and in taking advice from risk standards, such as [5] we need to

go beyond possible superficial risk mitigation strategies for improving safety of this particular application and deal with the core problem. This core problem is the use of a classifier algorithm that may provide a misclassification. How a classifier deals with points outside the boundary of our training data [6, 7] is at the core of how a classifier ultimately performs, either adequately or not.

Therefore, the use of an ANN that gives unpredictable results when presented with data outside the boundary of the training set, coupled with a safety critical application, such as powered wheelchair control can be considered a safety concern or problem. However, an obvious question that could be asked given the previous sentence is "why use an ANN in a critical application if it is a potential safety issue?"

The answer to this question, as simply as possible, is that the ANN has a desired characteristic called generalisation that allows the trained classifier to associate or generalise unseen patterns with a desired classification [8]. This characteristic, while desirable, becomes a "double edged sword" that is the source of the unpredictability and the problem being addressed by this research and thesis.

OUTSIDE THE BOUNDARY OF THE TRAINING SET

Literature discussing ANN generalisation tends to look at ways to improve the algorithm implementation with "tweaks" and other approaches, however the work by [6, 7] directly addresses the issue of generalisation and "outside the boundary".

This work is important as it attempts to highlight an issue associated with generalisation and what is an inherent limitation in the use of an ANN. However, the scope of the discussion becomes limited to simple two-dimensional examples and loses validity once real-world complex problems are encountered (as shown by this research).

The significance of this initial work is one of importance as the author has taken a critical assessment of what is a limiting factor affecting the use of an ANN. This limiting factor is significant as the concept affects not only an ANN, but other types of classifiers as well.

Taking a hypothetical situation, where you as the "patient" are dependent on a device that uses a classification algorithm to interpret some inputs and then control a "life giving" medication (for example). If the input space is large and potential inputs are

complex, would you as a patient trust your life to an algorithm that behaves unpredictably when presented with input data that is outside the boundary of the training set? A rhetorical question of course, but as a patient the expectation is that the issue of "outside the boundary" should not occur if safety is paramount. Yet, where an ANN is used, outside the boundary of the training set is an important limitation over its use, especially in safety critical applications.

1.2 Accuracy & Predictability

As discussed in the previous section the motivation for this thesis lay with the safety concern when using classifier algorithms such as an ANN in critical applications, especially when the algorithm is required to deal with data outside the boundary of the training set.

To enable a focused examination of this problem with resulting solution, this thesis is using head movement control of a powered wheelchair as the critical application of interest.

Note, that this particular technology was selected as previous work at UTS by others [2-4] resulted in a "hands free platform technology" (head movement), which became an area of interest as it uses an ANN to decode head movement.

TECHNOLOGY CONTEXT

Given the context of "hands free technology", this thesis is concerned with the performance of an ANN as used in a critical application, namely the control of a wheelchair using head movement. The technology context is shown below.

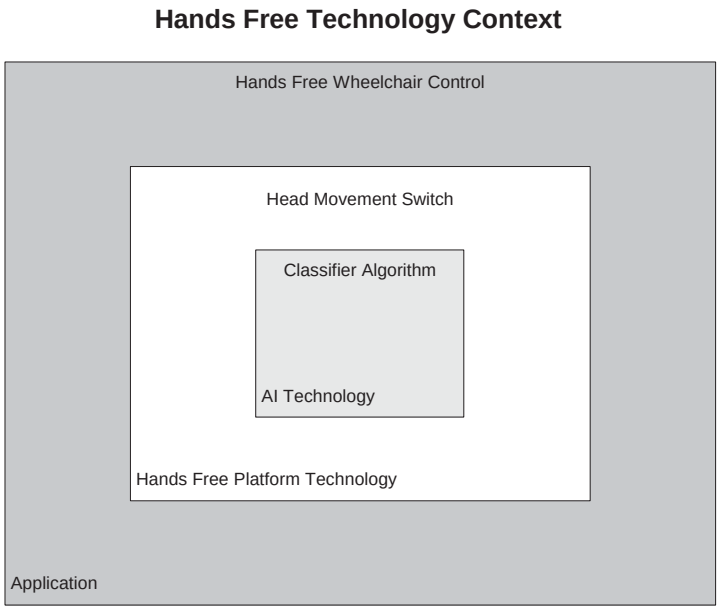

Figure 1-1: Hands Free Technology Context

Fundamentally, three levels of technology exist in realising "hands free wheelchair control". Firstly, the application itself, secondly the platform technology that allows the application to be realised and finally the artificial intelligence (AI) technology that is the "smarts" behind the platform technology.

In the context of "hands free wheelchair control", the application is the hands free wheelchair control, the platform technology is the head movement switch, and the AI technology is the classification algorithm. In this case the classifier algorithm is an ANN.

Within this context this thesis is addressing the problem of uncertainties that affect the accuracy performance (and hence safety) of an ANN as used in head movement decoding for control purposes of a wheelchair.

PROBLEM DEFINITION

Within the context as described and focusing on the ANN, the accuracy and predictability of the ANN are the characteristics that ultimately affect the safety of the wheelchair. These characteristics are significantly affected when the ANN is required to classify a point outside the boundary of the training set and becomes the focus of this research and thesis (see below).

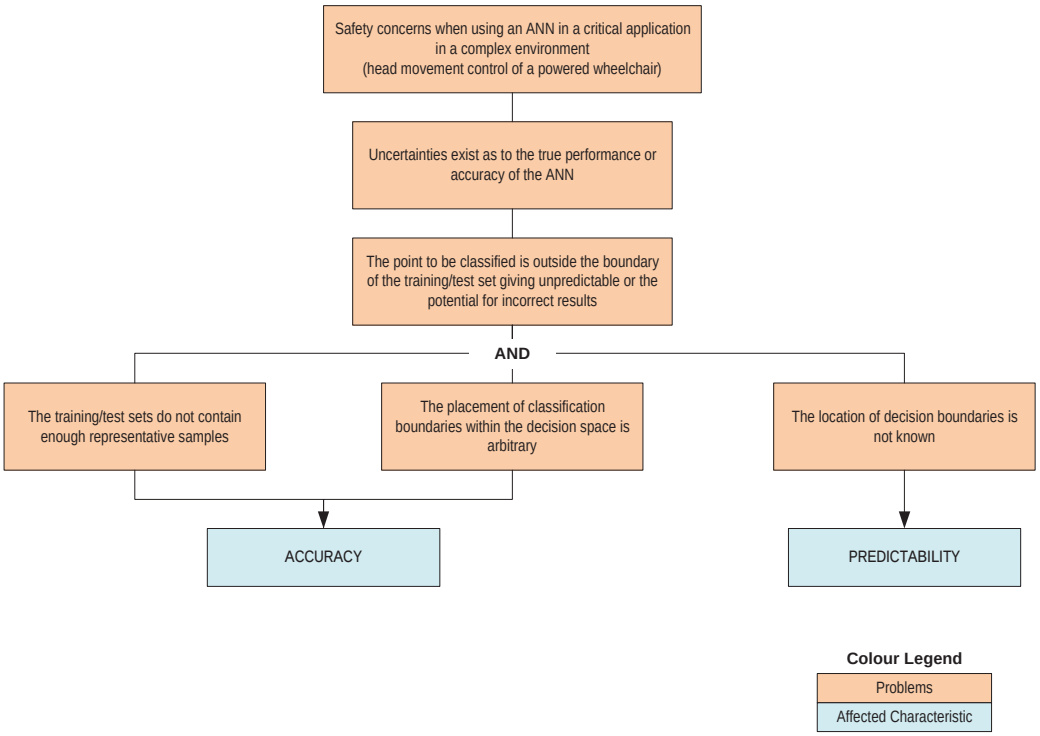

Figure 1-2: The Problem

Note that the concept of precision is implicitly covered by the concept of accuracy. A system can be "precise", but not accurate, where as an accurate system will have the necessary amount of precision to enable it to be accurate.

PROBLEM STATEMENT

In summarising, the problem underpinning this thesis can be stated as:

"How to ensure the accuracy and predictability of a classifier, used for head movement classification, for data points outside the boundary of the training set."

1.3 Aims of Research

Given the context as explained in Section 1.1 Motivation and Section 1.2 Accuracy & Predictability, the aims of the research consist of the following:

- To propose that the formation of implicit boundaries in an ANN through the addition of training data outside the boundary of the original training set provides improved performance when compared with an ANN trained with the original training set.

- To propose that the formation and use of explicit boundaries (Hyper-Rectangle) from original training data as a classification method provides improved performance when compared with an ANN trained with the original training set.

- To propose that the use of an Optimised Region and Boundary Classifier provides improved performance when compared with an ANN trained with the original training set.

- To propose that the use of an Optimised Region and Boundary Classifier provides improved performance when compared with an ANN or Hyper-Rectangle trained on the same data.

1.3.1 Contribution

The contribution that this research makes is:

- To provide a novel algorithm that addresses both the “accuracy” and “predictability” of a head movement classifier used in a complex environment to control a powered wheelchair.

- To provide a method to improve the performance an ANN as used to classify head movent based on the augmentation of the training data with “complementary” data (data either side of an implied boundary) from collected data.

- To overall enable the use of classification algorithms (for 2-D real time waveforms) such as an ANN and the developed novel algorithm to be used in applications that could affect safety. These applications are typically related to Medical Devices.

1.4 Outline of thesis

The layout of this thesis is described by the following:

- **Chapter 1 - Introduction:** Provides the motivation and context for the thesis with a stated problem. Also, the aims of this research are discussed and presented in this context.

- **Chapter 2 - Literature Review:** Provides a literature review of the problem and detailed analysis and discussion of the problems associated with Artificial Neural Networks. This chapter provides assessment of the current thoughts on how to address the problems associated with Artificial Neural Networks, generalisation, the arbitrary placement of boundaries, and training data. Finally, this chapter gives an overview of proposed conceptual solutions to address the problem.

- **Chapter 3 - Effective Head Movement Boundaries with an ANN:** Provides additional discussion, explanation, and interpretation of the problem and solution.

- **Chapter 4 - Bootstrap and ROC Analysis for Head Movement:** Provides information on a combined Bootstrap and ROC Analysis method used as part of the experimental methodology. Also, provides the description of the methods used to obtain data, the training used, the types of classifiers created, and the method used for comparing algorithms.

- **Chapter 5 - Implicit and Explicit Boundaries for ANN:** Provides the main findings of the research work, by showing how the formation of effective boundaries implemented either implicitly or explicitly are the result of representative data obtained with data outside the boundary of the original training set. This solution is compared to the baseline Artificial Neural Network trained with a standard training set (i.e. no additional data outside the boundary of the training set). Also, this developed solution is compared with some other strategies to identify the best performing method for the head movement control of a powered wheelchair.

- **Chapter 6 - An Advanced Boundary Concept:** Provides an improvement on the main finding of the research work, by explaining how an explicit boundary augmented with additional regions and boundaries and optimised provides a better solution to the problem. This improvement is called an Optimised Region and Boundary Classifier. Also, this developed solution is compared with the baseline Artificial Neural Network and other solutions/methods shown in Chapter 5 to confirm the improvements in performance.

- **Chapter 7 - Conclusion:** Provides an overall summary of the thesis with a discussion and conclusion. There is also discussion on the on significance of the research and other opportunities arising from this research are highlighted.

Chapter 2 - Literature Review

2.1 Problem Statement

Within the context of “hands free wheelchair control” as explained in Chapter 1 - Introduction. This thesis at a high level is addressing the problem of safety when using algorithms such as an Artificial Neural Network (ANN) in critical applications in a complex environment that use a head movement classifier (such as head movement control of a powered wheelchair).

By addressing this high level concern at the algorithm level it is clear that there are many potential problems related to the use of an ANN in head movement classification and these problems directly affect the accuracy and predictability of the ANN.

However as discussed, the ANN characteristically presents uncertainty as to behaviour when presented with data outside the boundary of the training set and therefore presents a safety problem when used in a critical application. Based on this, the problem statement becomes:

“How to ensure the accuracy and predictability of a classifier, used for head movement classification, for data points outside the boundary of the training set.”

2.2 Introduction

It is clear that a well constructed and trained ANN acting as a classifier performs very well on patterns that are close to those of the training set and in conditions that are essentially controlled, and in these circumstances is known as a “universal approximator” [9]. Once the ANN is required to perform a classification on data “outside the boundary” of the training set, problems with misclassification can result [6, 7]. Hence there is an uncertainty regarding the generalisation ability of the ANN. This problem was nicely expressed by [6] as "Making a Multi-layered Perceptron network say - "do not know" when it should", and encompasses one of the problems of generalisation.

Ideally, a ANN classifier should be presented with training data, learn the underlying function provided by the training data, and provide generalisation of this learned function to other data not part of the training set. In other words data not part of the

training set, but belonging to the class represented by the training data should be classified as such by the ANN. This characteristic is shown ideally in the figure below, in which the trained classifier (for one class) "generalises" past the boundary of the training set and correctly incorporates members of the same class.

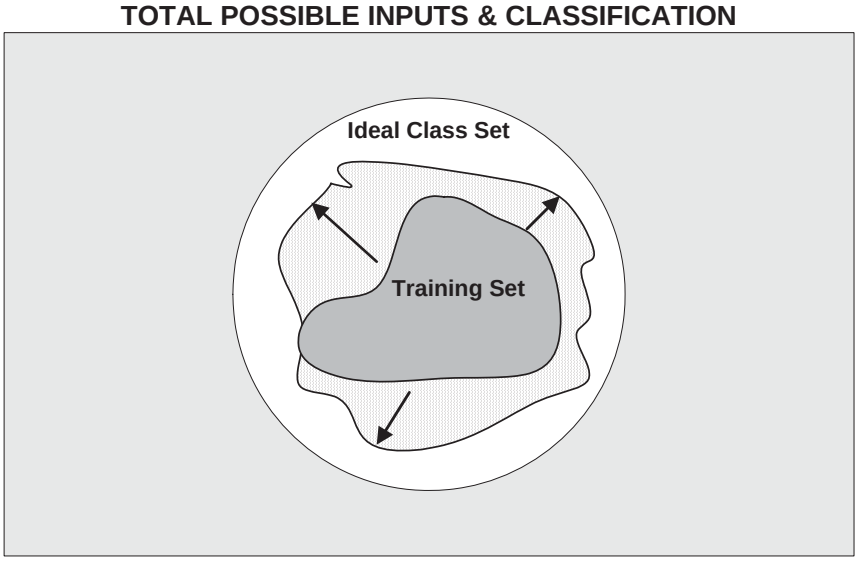

Figure 2-1: Generalisation – Single Class

In practice, this may not always be the case as there are numerous factors that can affect the generalisation ability of an ANN. From [10, 11], these can include:

- The training data itself;
- The complexity of the problem and the set of non-linear approximation functions required to represent the underlying function of the training data;
- The model architecture (feed-forward, feed-back, interconnections, number of layers, output combination, and number of hidden neurons etc);
- The model parameters (learning constant and activation functions, etc);
- The learning rules applied (Genetic Algorithm, Error Back Propagation, and other variations or optimisations);
- The starting point of training (initialisation of weights);

- The stopping point and stopping rules of training (e.g. use of validation set, minimum error and its value, etc); and
- Over-fitting and under-fitting, which are related to training data, learning, and stopping points, but are important factors never the less.

As we can see there are numerous factors that can affect the generalisation capabilities of an ANN, but even if all the factors above are controlled and implemented "correctly" there are still fundamental uncertainties associated with the use of these algorithms.

Fundamentally, without knowledge as to the correct membership of data points outside the boundary of the training set and some control over the boundary placements then uncertainty regarding the generalisation ability of a trained ANN will remain.

The Data or Algorithm?

To highlight which factor presents the most influence over the generalisation accuracy of an ANN, some discussion is required, as it can almost become a "chicken and egg" argument (which comes first?). So, is it the data or is it the algorithm?

Without doubt the training set is the starting point for the ANN to be able to generalise, but the other aspects related to the mechanics and detail of a ANN classifier algorithm also have influence over the boundary placement and hence generalisation abilities of the ANN. Bearing in mind, variations between boundary placement and hence generalisation performance (although small) can be expected between Neural Networks trained with the same data set due to the randomness of weight initialisation [12-14].

Let's suppose that we train an ANN on all possible inputs, then regardless of what we did with the algorithm, as long as the algorithm is used correctly we should be able to successfully train and correctly classify every single input. Of course there is no generalisation occurring here as we have used every single input in the training set, but the point is that variations in boundary placement due to the algorithm shouldn't affect the final accuracy.

Now let's suppose we reduce the number of training points; then the effects due to both algorithm and training set, with associated boundary placements will affect the generalisation performance of the ANN classifier.

In looking at both these cases there seems to be an obvious relationship between training points and certainty. The more training points we have the more certain we are about the output, and the fewer training points we have the more uncertain we are about the output. In practice most applications would present uncertainty, so clearly there are two major factors that influence the generalisation ability of an ANN, factors relating to the training data, and factors relating to the algorithm.

To complete this "chicken and egg" argument it would appear that both data and algorithm affect the boundary placement and hence generalisation performance of an ANN. However, reference to the old adage "garbage in garbage out" is given as a reminder that no matter how good the algorithm, the input data must also be good.

2.3 How Others Address the Problem

The approaches and methods proposed by others to address the problem of generalisation in Neural Networks are varied, but fundamentally are aimed at the effects imposed by the training data and those of the algorithm itself. The approaches proposed by others include:

- Artificially generated boundary data added to the training set [6, 7];
- Noise added to the training set [15-18];
- Creation of Virtual training examples [19];
- Bagging [20, 21];
- Boosting [22-25];
- Semi-labelled or unlabelled data added to the training set [21, 26-28];
- Randomised labelling of training data then added to the training set [29-31];
- Ensembles [25, 32-35];
- ARCING [36];
- Neuro-Fuzzy Hybrids, and those using hyper-boxes as boundaries [37-41];

- Boundary Feature Extraction [42];
- Neuro-Statistical Hybrids [43]; and
- Threshold Optimisation (ROC and cost functions) [44-52].

Augmenting Training Data with Artificial Boundary Data

An informative explanation regarding generalisation and uncertainty in an ANN is that this uncertainty is due to the random formation of boundaries with any given training set and how these boundaries relate to data "outside the boundary of the training set itself" [6, 7].

Once an ANN is presented with data outside the boundary of the training set, then inherently there will be uncertainty, as the formation of boundaries in relation to these points is to an extent random (due to the influences of the training set and the algorithm).

A way to combat this uncertainty is to add data outside the boundary of the training data to the original training set to help form boundaries. The now augmented training set would then influence where the ANN placed boundaries, hence controlling the generalisation and decreasing the uncertainty associated with generalisation.

Now, when the ANN is required to classify a point outside the boundary of the training set it should say, "do not know" instead of potentially incorrectly associating the data point with a specific class.

One method used to obtain training data "outside the boundary" is to use a hyper-cube or hyper-box expanded past the boundary of the training data. Points on this boundary are randomly selected and then added to the training data.

Of course previous work by others tends to be on simple 2-dimensional examples and as such questions were raised, such as how many points should be used and how far past the boundary of the training data should the boundary be place? However simple the examples, it provides a powerful insight into the problem and possible solution.

This type of work along with proposed solutions could be classified as the use of "artificially derived data" to augment a training set to increase generalisation certainty.

While providing an insight the previous work by others has problems, in that where should the boundary be placed once outside the training data boundary (expansion of the hyper-cube) and how many data points should be generated?

Both valid questions, but these could not be answered adequately as the example problems were artificial and hence there is no real knowledge or reason as to class membership once outside the boundary of the training data. As for how many training points are required, again as examples are artificial (2-dimensional) the number of required points to form a boundary is not a limiting factor, therefore any problems in data size were not realised and hence the question.

Realistically, there are two limiting factors on the number of data points outside the boundary to be added to the training set. One is the number of points in the class or training set itself, where the number of "non-class" points should ideally be equal to the "class" points [53]. The other is the number of points required to form a credible boundary, which would be all points on the boundary at least one point wide. In a simple 2-dimensional problem (as given by others) there are no real limitations on the potential size of the data set, however in a 40-dimensional problem with 256 possibilities per dimension the number of required points is potentially so large that it cannot be realised.

However, if we were to take a lateral thinking approach to the problem of creating a boundary out of data points on a hyper-cube boundary, then why not use the hyper-cube itself as the boundary, rather than selected points? In this instance the hyper-cube becomes a basis function or kernel function that allows classification of an input point to occur. However, we are no longer using a Multi-Layer Feed Forward Perceptron Artificial Neural Network Classifier, but a solution none the less.

The idea of using hyper-cubes or hyper-rectangles, which are easily described by minimum and maximum values per dimension has been previously proposed by others including [37, 39, 54] for various types of classifiers (Radial Basis Functions and Neuro-Fuzzy Hybrid Classifiers for example). Each of these provides a means to provide a known boundary, but note that this doesn't necessarily mean improved generalisation.

Interesting, Neuro-Fuzzy Hybrid Classifiers [55], could be seen as a natural progression in answering the problems of generalisation discussed in [6, 7], in which the idea of boundaries are raised. Generally speaking, the use of a Neuro-Fuzzy approach can be seen as an attempt to impose rules over the training data, which is another way of creating or imposing a boundary.

A brief comment on the issue of why creating a boundary such as a hyper-cube may not improve generalisation, is required. When creating the boundary, points outside the boundary will be rejected and points inside the boundary will be accepted. In a way we are limiting the generalisation to that close to the training data, which is the intention, but points outside the boundary that we want to associate with the class will now be rejected.

If we use some terminology borrowed from medical testing and diagnosis, namely, measures of sensitivity and specificity [56] and [57], we could say that the use of a boundary affects both sensitivity (those points part of the class and classified correctly) and specificity (those points not part of the class and classified correctly), due to the generalisation characteristic being constrained by the boundary.

Note, that the use of an explicit boundary is a valid way to provide known boundaries.

Augmenting Training Data with Noise

The addition of random noise to the training patterns is considered in [15-18], and is an interesting concept. The process of adding noise or "corrupting" the original pattern allows the creation of another pattern to add to the training set. By doing this, boundaries of the original training set are expanded, which could be seen as improving generalisation. Also, this approach would help alleviate problems associated with over-training or over-fitting.

Whether this approach would have significant effects on generalisation accuracy is open to debate, as we could say that the addition of noise pushes the boundary of the training data and could potentially improve the sensitivity of the ANN, but without attention to boundaries to constrain the expansion, make the ANN less specific. Therefore gains in generalisation accuracy may be minimal.

Note, that this method could be applied to the training data to create a validation set to again help with over-fitting. This removes the need to separate the training data into two sets, which reduces the number of samples available for training.

Augmenting Training Data with Virtual Examples

The creation of virtual training examples as given in [19] and related to a face recognition application, is another attempt to augment the training data to help with generalisation.

In this example, the same pattern is manipulated or transformed without altering the basic "features". Simple transforms such as symmetry are exploited to make a more robust training set, to again improve generalisation and help alleviate effects of over-fitting.

Bagging

The concept of "bagging" (**b**ootstrap **agg**regat**ing**) [20, 21], is to take an original training set, create multiple "bootstrap" training sets, train multiple classifiers, then combine the classifiers and by way of a majority vote (for example) obtain an output.

This method can be categorised as a variation on the ensemble or committee classifier approach, where numerous classifiers are combined and a majority vote taken as to the classification of a given input.

The foundation of why bagging is used is related to the variance experienced between classifiers trained on the same data, and to the variance due to effects imposed by the random partitioning of training, validation, and test sets [58, 59].

By combining the classifiers the effects due to the algorithm can be alleviated. Furthermore, this method allows the use of all collected data to be used for training and testing without introducing bias due to the use of non-independent test sets.

Typically, when creating an ANN for an application the data will need to be partitioned into training, validation, and test sets with a trained classifier the output. Now, this classifier has not been trained on all the data, so while it may produce an accuracy measure on the test set, if the same three sets were partitioned differently a different

measure would be achieved. This variation can be alleviated by way of bagging, and can improve generalisation performance over a single classifier.

Again, this method does not control the generalisation or the effects of the training data as such, but is a method to cope with the effects of the algorithm and data partitioning.

Boosting

Boosting is a term used in the context of "boosting the performance of a weak learner or algorithm" [22-25]. Therefore, this term is used quite interchangeably when describing any method to "boost" performance of a classifier.

From [60], boosting consists of creating a classifier with an accuracy better than average and subsequently adding classifiers to form an ensemble. With each successive classifier partially trained on the data points incorrectly classified by the previous one. When the classifiers are combined an improvement in performance is expected, hence the classifier performance has been "boosted".

There are variations or optimisations on boosting such as "AdaBoost" (Adaptive Boosting), which again helps to boost the performance and does so by continually adding "weak learners" until an accuracy goal is met [60].

While results of boosting are reported as very good on the examples given, however the idea is to start with a "weak learner" and gradually make each successive classifier better, but if you already have a high accuracy from the "weak learner" the effects of boosting will be minor or negligible [60].

Semi-Labelled Training Data

An example of an interesting approach to issues of generalisation is the concept of "semi-labelling" or "unlabelled data" [21, 26-28]. Semi-labelling allows an originally trained classifier to label data and then this data is added to the original training set to create another classifier.

In the examples given one of the stated advantages of this method is to reduce the mundane task of labelling training data from large pools of collected data. Also, there

seems to be improvements in generalisation performance over those classifiers trained with only correctly labelled training data.

Although, the applications report good results, the potential for this approach to end in disaster is high. However, the fact that this approach may work is based on the fact that the original training data (correctly labelled) is still used to train the classifier.

Realistically, where this approach may be of possible use is in controlling over-fitting, a problem associated with training.

Randomised Labelling of Training Data

Again an example of an interesting approach to generalisation is the concept of randomisation [29-31], which could seen as a similar concept to "semi-labelling" [21, 26-28] (and is also known as "semi-supervised training"). In randomisation, extra training data is generated by randomly assigning class labels to training data and then including these in the original training set.

As with semi-labelling, the application reports good results, but the potential for this approach to end in disaster is high. However, the fact that this approach may work is based on the fact that the original training data (correctly labelled) is still used to train the classifier.

Realistically, where this approach may be of possible use is in controlling over-fitting, a problem associated with training.

Ensembles and Committee Classifiers

The use of combinations of classifiers either of the same or different types to essentially "vote" on a classification is the domain of ensembles [25, 32-35].

As with bagging and boosting, classifiers are combined after being trained on different training sets and a majority vote taken as to the classification of any given input. However, bootstrap samples of training sets aren't used, but a cross-validation method is used to help create the classifiers.

As with bagging, this helps to alleviate the effects of the algorithm and data partitioning.

ARCING

ARCING [36, 60] (Adaptive Reweighting and Combining) is realistically a general term for reusing and selecting data [60]. Methods such as Bagging and Boosting (AdaBoost for example) are considered as ARCING.

Neuro-Fuzzy & Hyper-Boxes

The use of hyper-boxes as a means to classify input data or to serve as the basis for other classification algorithms such as Radial Basis Functions or Neuro-Fuzzy Hybrids [37-41] is an interesting approach.

The hyper-box or hyper-cube is easy to implement, as all that is required is the minimum and maximum value for each dimension. From this, simple class memberships can be obtained.

The use of such a function provides an attempt to impose a known boundary around a grouping or cluster of data. While this can be successful and produce good results, there are possible trade-offs. In creating such a boundary improvements in specificity may be achieved, but the generalisation is now constrained and the sensitivity of the classifier may be affected with subsequent impacts on performance.

The use of such functions, applied to classifiers such as these Neuro-Fuzzy Classifiers is seen as an attempt to impose bounds on generalisation and hence "control" generalisation, in a "fuzzy" sense.

Boundary Feature Extraction

The idea of Boundary Feature Extraction [42] is included as it is also an attempt to find a boundary which describes the class memberships. Once this boundary is discovered then classification can be achieved.

The main points of the example given here is that the ANN is used to find the boundary between classes after being trained. Once the boundaries are found a feature vector is developed, and then this is used to create a "simpler" ANN.

A novel idea, but it is not addressing the issue of generalisation, is rather to use the ANN to create a more streamlined ANN architecture with faster response times.

Neuro-Statistical Hybrids

Another example of a method to find boundaries is to find the underlying probability distribution function from which to select training examples and then train a ANN (Neuro-Statistical Hybrids [43]).

This is not a new approach, but the novelty here is that information theory is used to find the boundaries between classes.

Again, this method is a novel idea, but it does not address the issue of generalisation.

Threshold Optimisation, ROC, & Cost

When using a Continuous Perceptron Classifier the outputs are generally passed through a threshold to give a discrete output value of 0 or 1. This threshold can impact the generalisation performance of a classifier and one means of improving this is to move the threshold to obtain the best accuracy.

Using measures of sensitivity and specificity [56, 57] the Receiver Operating Characteristics (ROC) can be generated to assist in evaluating the given classifier and in determining the optimal threshold value for best performance [44-50].

Although, used extensively in medical testing and diagnostics, this method of evaluation and optimisation is recommended.

Associated with this optimisation method is the concept of "cost" or "cost function" [50, 52] associated with misclassifying an input to the ANN. A basic example, commonly given is in cancer detection and imaging, where to incorrectly classify someone as not having cancer when they do (false negative), may literally mean the difference between life and death. In this situation the "cost" or "consequence" of the misclassification is higher than incorrectly classifying someone as having cancer when they do not (false positive).

Although, the example is simple the concept of the "cost" can help to establish an optimal threshold where the weighting given to measures of sensitivity or specificity is adjusted accordingly.

2.3.1 Discussion

In Section 2.2 the idea of the "Data or the Algorithm" was presented to help steer the literature review and the interpretation of what others were doing. In doing so, the review of others' work, begged the question "what are they addressing?" Is the work addressing the root cause (effects due to the data) or are they addressing a symptom (effects due to the algorithm)?

Interestingly, there has been quite a bit of work done to address the effect due to the algorithm, with "Bagging" or "Ensembles" seen here as the main contributors. Bagging however could be seen as the most promising way of addressing the effects due to the algorithm, as it allows the creation of a classifier (an Ensemble) while still being able to use all the collected data.

Realistically, Bagging could be very useful in producing a final ANN classifier for an application, as long as any real-time constraints, including space are met.

Other work, which addresses the effect of the algorithm, concentrates on "corrupting" the training data to ensure over-fitting is minimised. Methods such as semi-labelled data or randomisation of training labels are all attempts to stop over-fitting and hence improve the generalisation performance.

These methods could be considered "automated" ways of assisting the training process to limit the possibility of over-fitting, but caution is recommended with their use. Note, that there are other ways to address over-fitting, such as the use of a validation set or properly set error targets.

The addition of noise to a training set is also another method to combat over-fitting and would be a recommended method along with a validation set. A variation of this would be to add noise to the training patterns for the purposes of creating a validation set. Again, either method would be preferred over semi-labelling or randomisation.

Work on boundary formation and attempts to find the underlying functions described by the training data are also being undertaken, but seem to be used to build different types of classifiers.

Of the literature reviewed the idea of improving generalisation and certainty in an ANN by directly influencing boundary formation with augmented training data appears to be the most promising. Although, given simple artificial examples the problems posed were not adequately solved and in some cases additional work pointed to evolved solutions with different classifier types altogether (Neuro-Fuzzy Hybrid Classifiers).

A point to make is that the use of Neuro-Fuzzy Classifiers could be seen as an attempt to impose rules on the training data and process, and hence control boundary formation (an attempted solution to the original problem).

So, in trying to understand what others are doing in the context of "the data or the algorithm", there seems to be a large amount of work in trying to nullify the effects of the algorithm, but not much work in trying to address the input data issue. This has also been highlighted as a problem by [11] in a recent literature review, and could be seen as a "gap in knowledge".

Also mentioned in the literature review by [11] is the lack of research into the issue of non-stationarity, which refers to the use of classifiers on patterns or input data that is not stationary. An example of a non-stationary signal is one which is time varying, such as the head movement signal from transducers, used for wheelchair control. However, the issue of "non-stationarity" isn't the focus of this thesis.

In concluding this discussion, it appears that there is a gap in knowledge on how to influence the formation of boundaries to improve the generalisation performance of an ANN with the input training data (to create effective boundaries). Also, there appears to be limited literature on the problem of data outside the boundary of the training set and the uncertainties that it brings.

Note that one particular gap in knowledge is how to use real data to augment the existing training data to influence boundary formations. Most of the papers are aimed at the generation of "artificial data" to augment the training sets.

2.4 Representative Data & Effective Boundaries

In providing a solution to the problem as stated in Section 2.1 and interpreted in Section 3.2 a helpful start is to take another look at the problem tree and identify some contributing solutions that conceptually address the problem (see below).

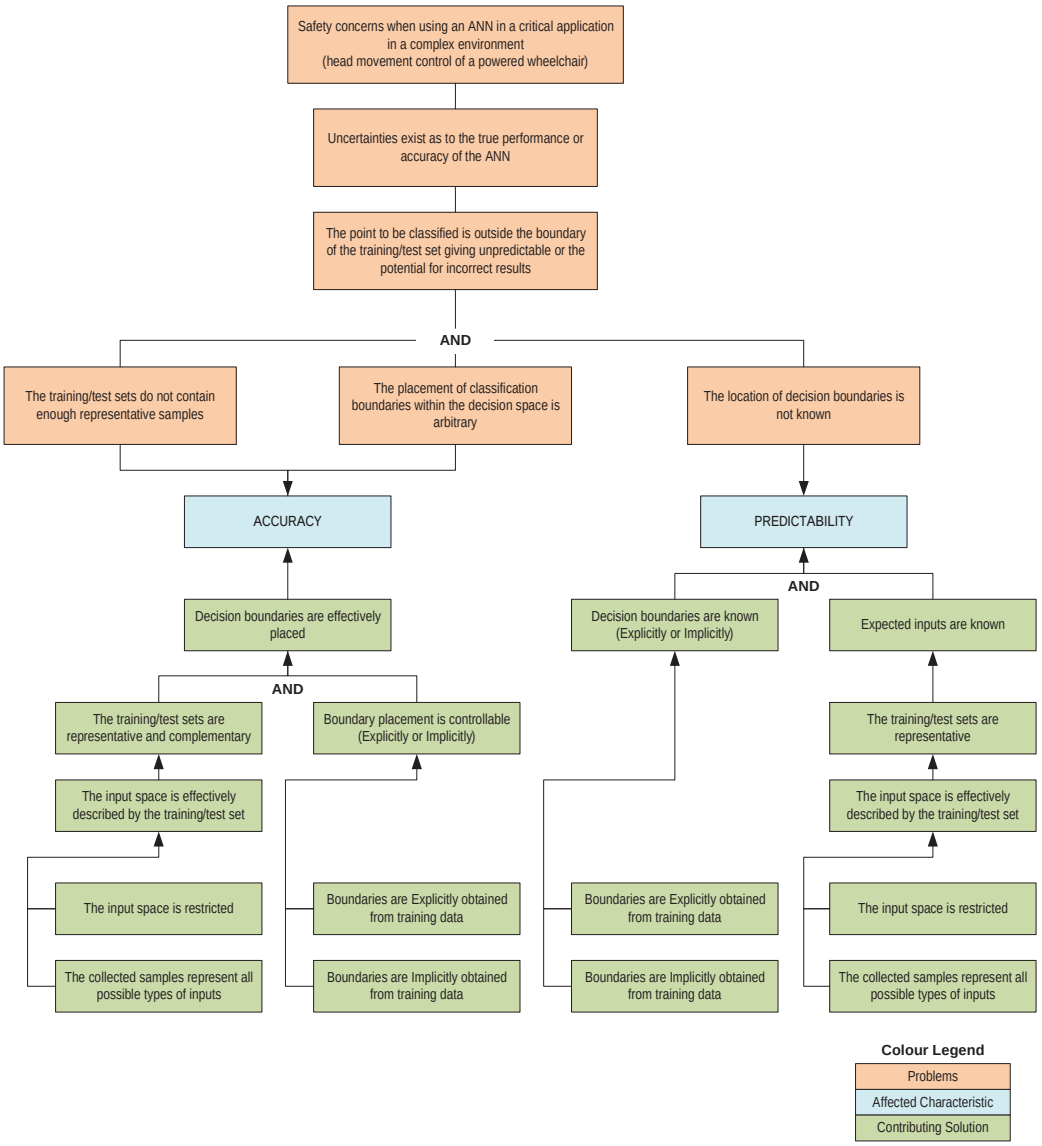

Figure 2-2: Problem/Solution Tree

Examination of this diagram shows that accuracy is dependent on the effective boundary placement within the decision space and predictability is dependent on knowledge of boundary placement and expected inputs to the classifier.

Interestingly, in finding a solution to accuracy, by ensuring the training/test set is representative and the boundaries are explicitly or implicitly defined, a solution to predictability is also obtained.

So the conceptual solution to our problem as provided by the problem tree is to have effectively placed decision boundaries that are known, which is reliant on the following:

- Have a training/test set that is representative and complementary; and
- Use boundaries that are either implicitly or explicitly defined.

Superficially, having a representative training/test set is all about having obtained all possible types of patterns that are likely inputs to the classifier. However, as boundary formation either explicitly or implicitly is dependent on the composition of the training set, the training set needs to be "representative enough" to enable effective boundary formation.

This would require that the training set be more than a collection of "likely" or "types" of input expected to be classified. It would require that the training data also be complementary i.e. data either side of an implied boundary that would then enable effective boundary formation.

2.4.1 Proposal

Given the stated problem (see section 2.1) and conceptual solutions the following is proposed:

- To create training sets that enable effective implicit and explicit boundary formations in an ANN and a Hyper-Rectangle Basis Function (as a classifier);
- To create explicitly defined regions and boundaries from training data to classify head movement (Hyper-Rectangle Basis Function, as the classifier);

- To optimise the explicitly defined Hyper-Rectangle Classifier using ROC methodology; and
- To create an improvement over the Hyper-Rectangle Basis Function (as a classifier), by augmenting the Hyper-Rectangle Basis Function with additional regions and boundaries to create an Optimised Region and Boundary Classifier.

Given the problem statement the following hypotheses are proposed:

- That the addition of data consisting of likely inputs to the classifier, but outside the boundary of a normal class training set, makes the training set more representative and complementary.
- That the addition of data consisting of points randomly selected on the boundary of a Hyper-Rectangle, but outside the boundary of a normal class training set, makes the training set more representative.
- That the use of an explicitly defined boundary such as that obtained from an Optimised Hyper-Rectangle is a valid method for classifying data.
- That an Optimised Hyper-Rectangle Classifier can be improved by the addition of regions and boundaries to form a better classifier known as an Optimised Region and Boundary Classifier.
- That an explicitly defined boundary performs better than an implicitly defined boundary when trained on the same data.
- That the Optimised Region and Boundary Classifier is the highest ranked classifier when compared to an Artificial Neural Network or an Optimised Hyper-Rectangle Classifier.

2.5 Discussion and Conclusion

In conducting this literature review, an overall problem was established and stated as:

"How to ensure the accuracy and predictability of a classifier, used for head movement classification, for data points outside the boundary of the training set."

As this problem was analysed further the major contributing factors to this problem were the representativeness of the training/test sets and the formation and knowledge as to the decision boundaries either explicit or implicit. These contributing factors directly affect the accuracy and predictability of the ANN classifier and are required to be addressed by this thesis.

During discussion of the problem the ANN characteristic of "generalisation" was raised as a desired characteristic of the ANN, but this can be a double-edged sword as there are also negative consequences.

In examining the issue of generalisation, the effects on performance were due to a range of factors, which was further refined to be the result of the effects of either the training data or the algorithm. In doing this, algorithm dependent factors could be separated from data dependent factors.

Having categorised factors influencing generalisation into data or algorithm dependent, a literature review was conducted to find how others have addressed the effects on generalisation due to these factors.

Using [6, 7] as the major starting point for data dependent influences, others such as [20, 29, 36, 61] offered other insight and methods to alleviate algorithm dependent influences on generalisation.

Of significant importance while defining the problem and understanding the work of others is the apparent lack of work on the effects of the data on generalisation, as backed up by [11] in a recent literature review.

In going back to the stated problem, what is important to note is that the classifier needs to deal in a correct and predictable way with data points outside the boundary of the training set. For this to occur it is proposed to create effectively representative and

complementary training/test sets and to use this data to create effective implicit and explicit boundaries.

These boundaries will be realised in an ANN and an Optimised Region and Boundary Classifier, which will address the stated problem.

However, only the Optimised Region and Boundary Classifier will be able to fully address the problem, as the ANN will always have unpredictability associated with it. It is this Optimised Region and Boundary Classifier that is presented as a novel solution that ensures that data outside the boundary of the training set is dealt with correctly and predictably.

Chapter 3 - Effective Head Movement Boundaries with an ANN

3.1 Introduction

This chapter discusses and interprets the problem and solution concepts in more detail then was previously mentioned and discussed in Chapter 1 - Introduction and in Chapter 2 - Literature Review.

As was briefly mentioned in Section 1.1, if an ANN is potentially a problem, then why is it used? The answer to this is that the ANN provides a non-linear mapping of inputs to outputs with the characteristic of generalisation. This very characteristic makes the use of an ANN desirable, but this characteristic can be considered a double-edged sword, which has desired and undesired consequences.

The desired consequence of generalisation is that points not part of the training set are correctly classified (see Figure 3-1: Desired Generalisation Ideal), with the undesired consequence of generalisation being the incorrect classification of such included points (see Figure 3-3: Over Generalisation Ideal).

Artificial Neural Networks and Generalisation

Generalisation capabilities of classifiers and in particular neural networks are a desired characteristic as this allows the trained classifier to associate or generalise unseen patterns with a desired classification [8].

As shown below in a simplified ideal, the decision boundaries formed during training go beyond the training set, but are still contained within the idealised notion of the ideal class set of which the training set is representative. This “expansion” outside the training set is called generalisation and allows unseen data “close” to the training set to be associated with that class. Although idealised this is the characteristic of the ANN that is desirable.

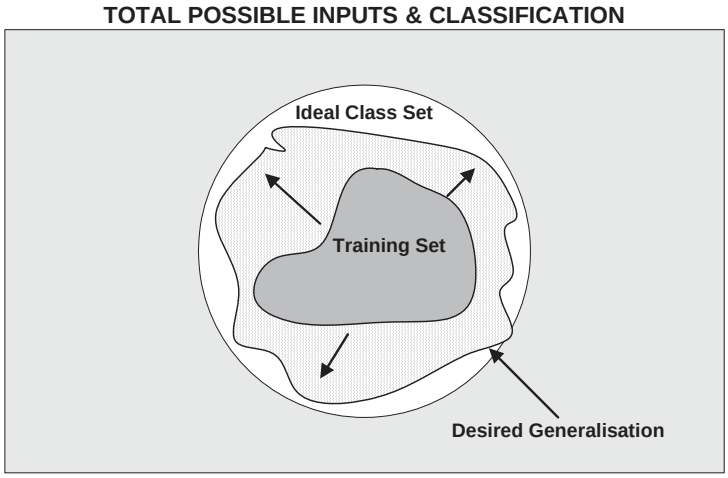

Figure 3-1: Desired Generalisation Ideal

To further highlight this concept a simple 2-dimensional example is shown below that contains four classes, which is the same number of classes for the head movement classifier used in this thesis.

As can be seen, those points not included in the training set, but next to each training point are included in the decision boundary for that class. This is "generalisation" in action and is considered as an important characteristic, but is also considered as a double-edged sword.

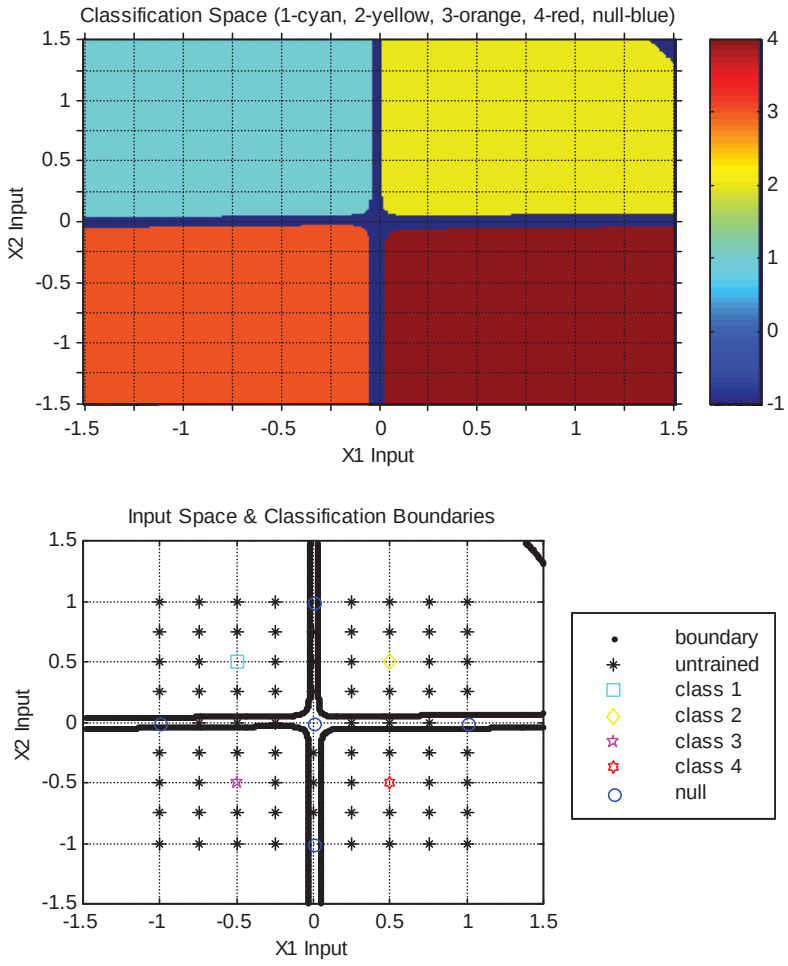

Figure 3-2: ANN Trained with 4 Classes (2-Dimensional Example)

In the above example, no information exists as to the set membership of those patterns not included in the training set. The fact that the ANN has set the decision boundaries and will classify those points as such does not mean that the classification is correct.

As an example it clearly shows that generalisation will include those patterns near the training data, but not in the training set in the formed decision boundaries.

However, there are other possibilities, which are most likely, such as the generalisation being too broad ("over generalisation") or too narrow ("under generalisation"). With over generalisation, the decision boundary is formed beyond the boundary of the Ideal Class Set and includes other data points that are not meant to be included in that class.

This type of generalisation or "over generalisation" would be considered as undesired in an ideal sense, as the classifier would incorrectly classify data (unseen data) outside the Ideal Class Set.

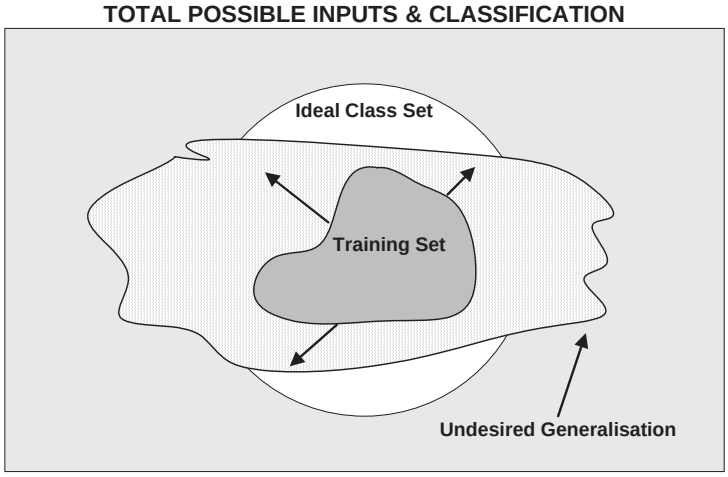

Figure 3-3: Over Generalisation Ideal

(Note that the terminology "over generalisation" is also known as "under fitting", with "under generalisation" known as "over fitting".)

If the decision boundary was to only include a small percentage of the points outside the boundary of the training set, but still part of the ideal class set the generalisation would be termed as “under generalisation” (see below).

In this case the classifier would incorrectly classify points in the ideal class set, which are outside the decision boundaries created with the training set.

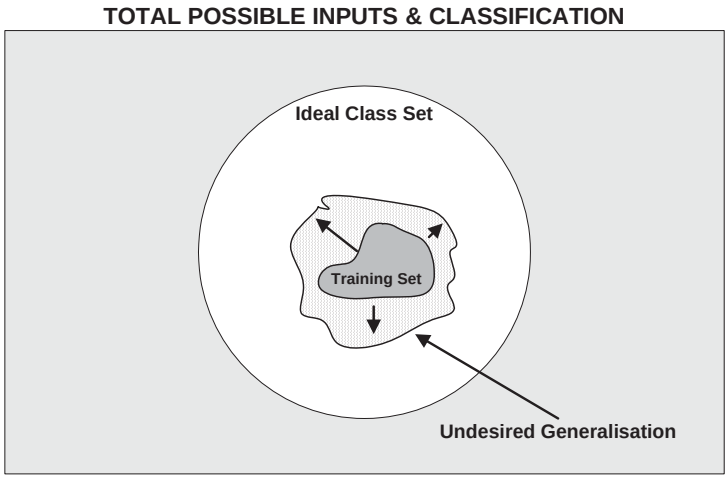

Figure 3-4: Under Generalisation Ideal

In examining the previous diagrams there are two significant unknowns, the Ideal Class Set boundary and the final decision boundary obtained after training. Both of these boundaries would remain unknown, unless explicit or implicit information existed as to their location.

The effect of generalisation in an Artificial Neural Network could be described as “partially controlled randomisation of boundary placement based on the training set” and this characteristic of arbitrarily placing decision boundaries provides the most uncertainty. This uncertainty is a problem especially when used in critical applications such as head movement classification (used in control of a powered wheelchair).

3.2 ANN Performance

Given the context explained in Section Chapter 1 - Introduction, this thesis is concerned with safety due to the uncertainties affecting performance accuracy of an Artificial Neural Network (ANN) as used in head movement classification (for hands free wheelchair control).

By addressing the uncertainties affecting the performance of an ANN, it is expected that any solutions discovered and/or developed will provide a higher level of confidence in the safety associated with the use of such an algorithm in this application.

To assist in identifying sources of uncertainty a problem tree was developed using a technique known as “five whys”, where questions are asked until a root cause is discovered. Typically, it takes at least five levels of questioning until a root cause or source is reached (hence the five whys). The figure below shows such a problem tree.

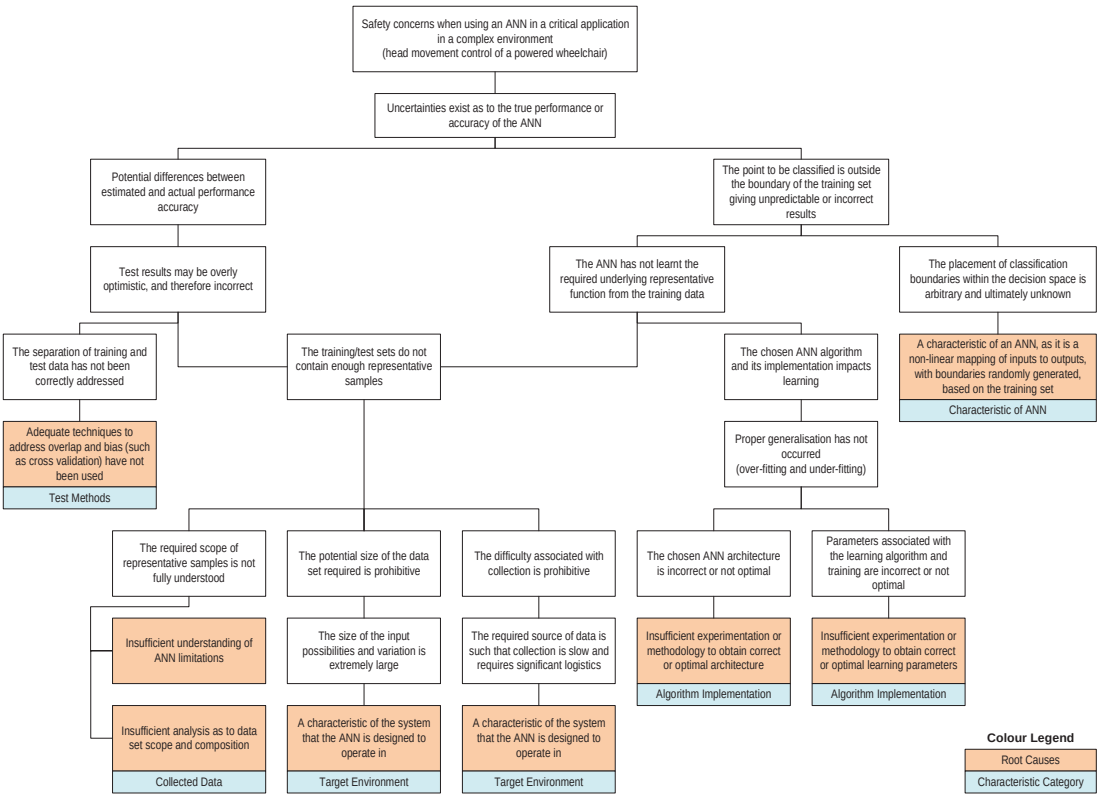

Figure 3-5: Problem Tree

In understanding the problems associated with the use of an ANN, there are two main sources of uncertainty affecting performance accuracy of an ANN:

- The potential differences between the predicted or estimated performance and actual performance; and
- The unpredictability of performance of the ANN when presented with data outside the boundary of the training set.

3.2.1 Estimated and Actual Performance Discrepancies

The estimated performance of an ANN depends on test results, which are used as predictors of future performance. Obviously, the better the test result the greater the confidence in actual performance. However, if the testing methodology is not thorough or adequate, or the data used to test with is not adequate, then the test results will not be good indicators of actual performance.

In looking at test performance two major contributors are obvious, the test methodology itself and the test data. There are various ways to address the test methodology and these tend to be variations on cross-validation (refer to Section 4.2 for further information).

Regarding the test data, regardless of the test methodology and the segregation of training and test data, if the collected data is not representative of the types of patterns the classifier would be expected to see and classify, then the test data is considered inadequate. This inadequacy will ultimately lead to overly optimistic results and hence is a source of discrepancy between the estimated and actual results.

3.2.1.1 Representative Data

Whether or not the training/test data is adequate fundamentally depends on the number of representative samples that are collected. The key term here is “representative”, which implies that the collected patterns contain samples of all the types of patterns that would be likely inputs to the classifier during operation (i.e. representative samples).

However, obtaining representative samples is dependent on a number of factors including:

- The definition of the types of representative data expected and required;
- The size or amount of required representative data; and
- The collection method and difficulty.

Failure to adequately understand the different types of patterns that the classifier could see in operation will reduce the coverage of representative samples and cause discrepancies between the predicted and actual performance estimations. This failure could be seen as an implementation failure where the problem space and required scope of the training/test data isn't fully understood. However, this particular problem becomes a fundamental problem with the training/test data.

The figure below highlights the idea of representative samples. Within the total input space there are three sets represented, the Ideal Class Set, the Likely Inputs Set, and the Collected Data Set. As represented below, the collected data is only a small subset of that which is likely to be seen by the classifier and does not include all the "types" of data. Therefore the collected data could be seen to be under representative of the likely types of inputs.

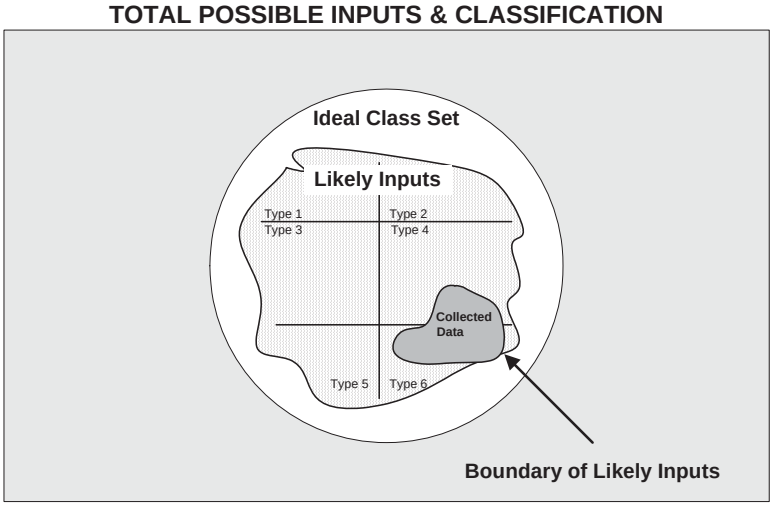

Figure 3-6: Under Representative Data

However, if the collected data were truly representative, then it would contain samples of all the likely types of data likely to be seen by the classifier. This ideal is represented below.

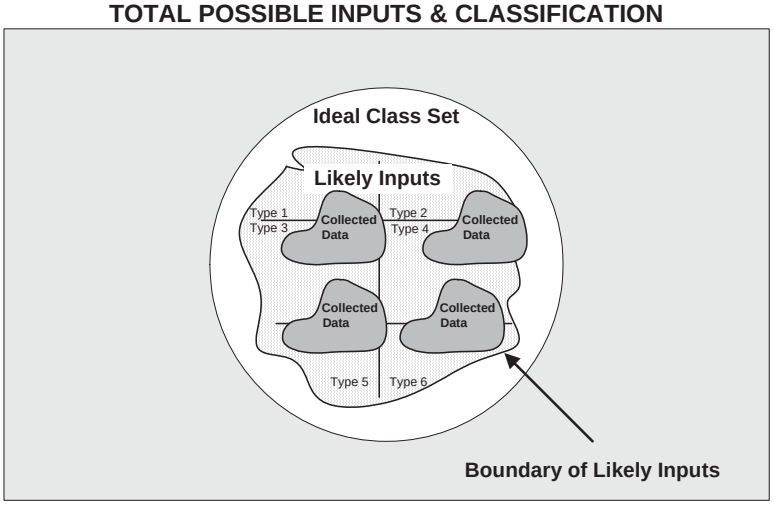

Figure 3-7: Representative Data

Other problems associated with the training/test data are the result of the target system or environment, which places constraints on the number of representative samples possible through physical means. One constraint is that the sheer number of required representative samples is so large as to be impractical to either collect or use. For example, if an input space has 256^{40} possible inputs (as in the case of this wheelchair application), then the required number of representative samples could potentially be quite large and impractical to obtain or use.

Also, the difficulty associated with the method of collection may be such as to limit how many samples could be obtained. In this wheelchair application the collection of samples was slow and difficult, which placed constraints on the total number of representative samples able to be collected.

In summarising the problems associated with predicted and actual performance, it can be stated that test methodology has an impact, but of more significance is the collected data and its representativeness. The differences between the collected data used to train and test a classifier and the likely inputs to a classifier during use can be seen as the biggest single factor affecting the predictive capability of any test. It must be noted that

this is based on the assumption that the chosen algorithm has been correctly implemented.

3.2.2 Outside the Boundary of the Training Set

One of the largest factors influencing the uncertainty of performance accuracy of an ANN is that when the classifier is presented with data outside the boundary of the training set the results are unpredictable or incorrect. This is due to a number of factors including:

- The ANN has not learnt the required underlying representative function from the training data; and
- The decision boundaries of the ANN arbitrarily placed based on the training data are not known; therefore you do not know how the classifier will react to points outside the boundary.

It must be noted that an ANN may not learn the "underlying representative function", but may learn enough to be effective. This distinction is highlighted as it differentiates between the ideal and the practical. In an ideal sense if the ANN learns the underlying function, then it would be reasonable to infer that it has learnt the "Ideal Class Set", which may not be known or knowable. This inference is therefore not reasonable as an Ideal Class Set is in a practical sense undefined. However, in a practical sense the ANN may learn enough to address the "Likely Inputs Set" that the classifier may see. This is a reasonable inference, but only if the "Likely Inputs Set" is defined.

In learning the required underlying representative function from the training data there are a number of factors influencing this including:

- How representative the collected training data is; and
- Algorithm implementation and impacts on learning.

Regarding the training data, if the training data is not sufficient to represent the types of data that the classifier would see (the "Likely Inputs Set") then it is unlikely that the ANN will be able to learn the required underlying function (see Section 3.2.1.1 Representative Data).

Regarding the algorithm and implementation impacts, the use of an incorrect architecture, training methodology, and associated parameters can have significant impact on whether or not the ANN is able to learn the required underlying representative function. However, as so much is known about the ANN algorithm, learning, and its implementation this is not considered a significant source of uncertainty in this case.

One of the biggest contributors to unpredictability with an ANN is that the location of the decision boundaries is not known. Therefore, once the classifier is presented with data outside the boundary of the training set (or test set) the results are unpredictable. This is an inherent characteristic of the ANN to arbitrarily place decision boundaries based on the training data. As this characteristic is inherent it is something that cannot be changed, but if using an ANN is required to be "managed".

However, this fundamental issue of arbitrarily placing boundaries and then not knowing where the decision boundaries are is a major source of uncertainty over the accuracy performance of the ANN, especially when the ANN is required to classify all possible input data and not just a restricted subset.

3.2.3 Trimming the Problem Tree

In looking at the various problems or sources of uncertainty that affect the accuracy performance there are some sources of uncertainty that can be considered to be "managed" and not major contributors in this case.

With this in mind the ANN algorithm, implementation, and the test methodology are considered to be "managed" and not the major causes of uncertainty in regard to the accuracy performance of the ANN (although important). Also, those aspects that are inherent to the target environment or system, such as size of the input space possibilities and collection difficulty are considered "managed".

By regarding some of the contributing sources of uncertainty as "managed" those contributing factors that are the hardest to control or manage can now be identified.

Using this approach the problem tree can be trimmed to show only those areas that provide the major sources of uncertainty (see below). Note, that only two main causes of uncertainty remain after trimming the problem tree, the training/test sample

representativeness and the arbitrary placement of decision boundaries based on training data, which is ultimately unknown.

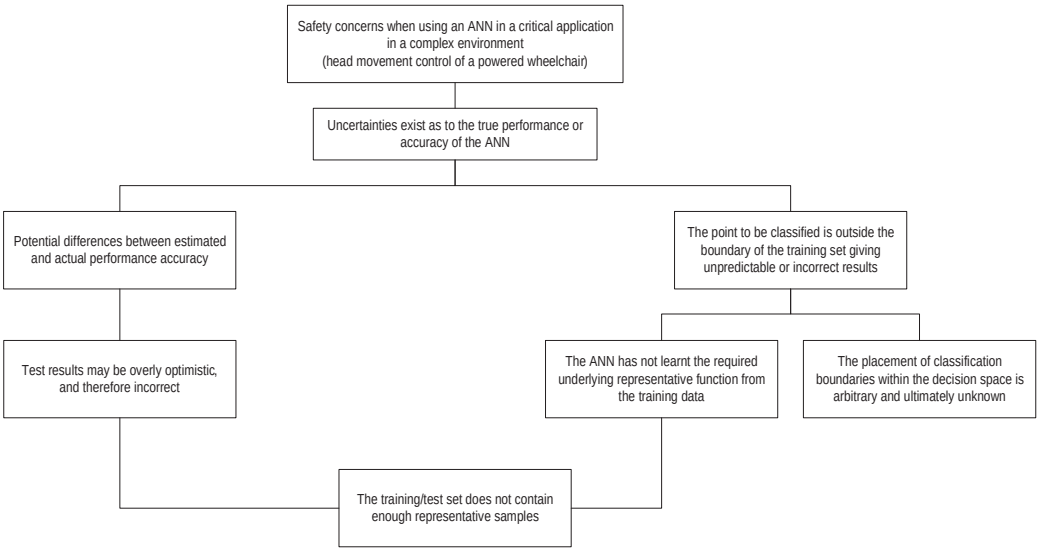

Figure 3-8: Problem Tree – First Pass Trim

An additional trim can be applied to leave the main causes of uncertainty visible in the problem tree. Again, as can be seen (below), there are clearly two major sources of uncertainty affecting the performance accuracy of an ANN, the training/test data, and the arbitrary placement of decision boundaries.

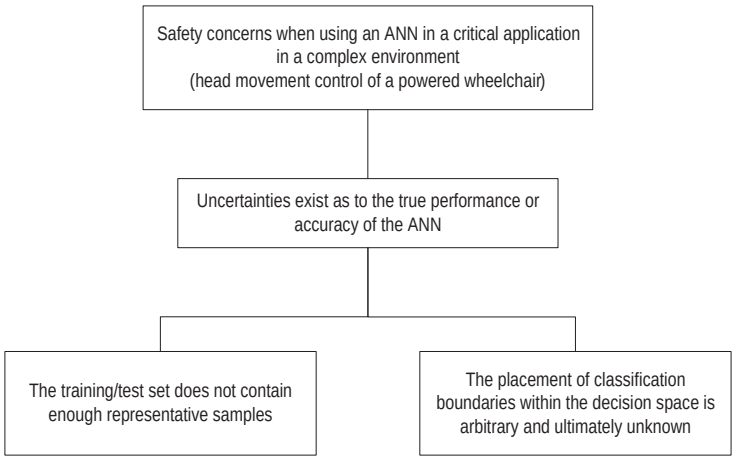

Figure 3-9: Problem Tree – Second Pass Trim

The problem can be further defined by examining why the arbitrary placement of boundaries would be a problem. Given a training set and correct training, the ANN would correctly classify data points in the training set, and some data outside the boundary of the training set, but close to this boundary due to the effect of generalisation.

However, as the boundary placement is arbitrary and unknown there are no guarantees that the ANN will correctly classify any data point outside the boundary of the training/test sets.

It must be noted, that knowing where a boundary is, doesn't affect the resulting accuracy of the ANN. Knowledge as to location of boundaries affects the predictability of the results, with the placement of boundaries affecting the results.

The effect of this is to create two problems, classification is unpredictable when presented with data outside the boundary of the training/test set, and potential accuracy performance is affected by the arbitrary placement of boundaries.

This is clearly unacceptable, however if the boundaries were formed in a controlled manner with resulting knowledge as to the location of such boundaries, then both problems would be addressed.

Having knowledge of boundary locations allows for the prediction of performance for any given data point and having control over the placement allows for the definition of how the classifier will perform for data inside and outside the boundary of the training set. Therefore, the classifier is no longer unpredictable, but to address accuracy the placement of boundaries must be correct!

For a classifier to be accurate (or correct) the training data must be sufficient to enable effective boundaries to be placed (see Section 3.2.1.1 Representative Data).

The impact of boundaries and training/test data representativeness on performance uncertainty and accuracy is shown in the table below. This table uses a relative qualitative scale comprising "Highest", "Medium", and "Lowest" to describe performance uncertainty and accuracy.

Table 3-1: Effect of Boundary Control, Knowledge, and Training/Test Data

Boundaries Known & Controlled (Y/N)	Training/Test Data Representative (Y/N)	Performance Uncertainty	Performance Accuracy
N	N	Highest	Lowest
N	Y	Medium	Medium
Y	N	Medium	Lowest
Y	Y	Lowest	Highest

As can be seen, the single biggest impact on accuracy is the training data and its representativeness, however, the biggest single impact on uncertainty is if the boundary is known and controlled. The combined effect when both are true is to provide the lowest uncertainty and the highest accuracy. The opposite is true when both are not present.

3.2.4 Redefining the Problem Tree

Given the identification of two factors, accuracy and predictability, with their contributing problems, a further redefinition of the problem tree can occur. This redefinition is shown in the Final Problem Tree below:

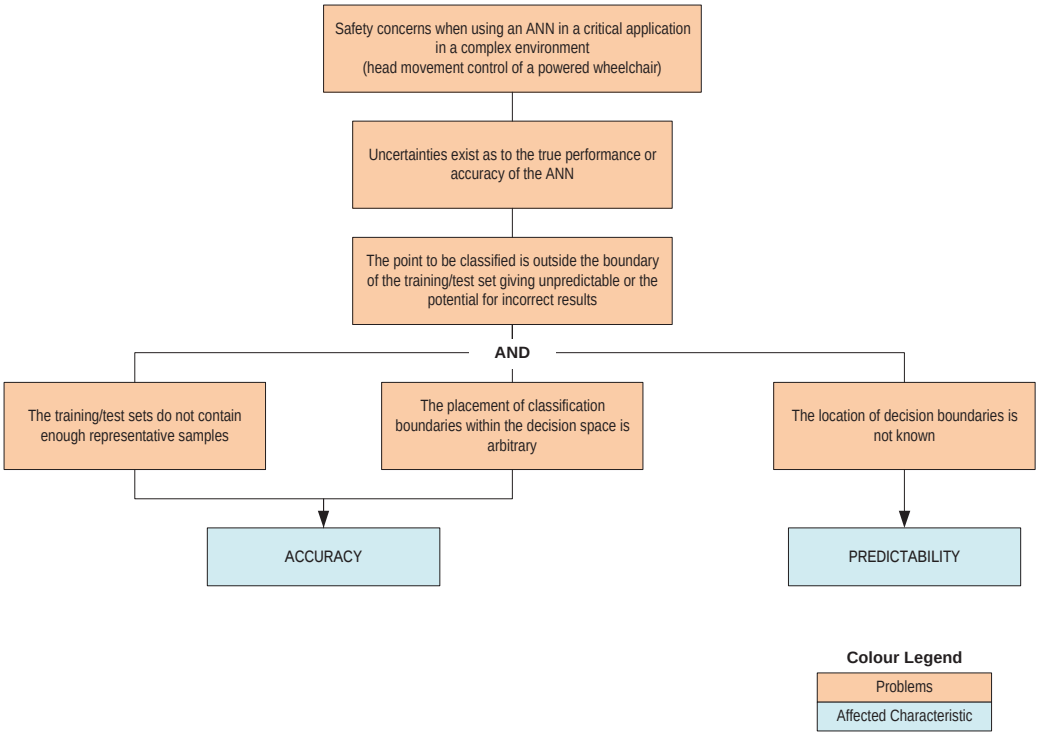

Figure 3-10: Final Problem Tree

Examination of this diagram shows that accuracy is dependent on the effective boundary placement within the decision space and predictability is dependent on knowledge of boundary placement and expected inputs to the classifier.

These problems directly affect the accuracy and predictability of the ANN and are the cause for the unpredictable or incorrect results when the ANN is presented with a data point outside the boundary of the training set.

3.2.5 Summary and Problem Statement

As previously discussed and mentioned in Chapter 1 - Introduction and in Chapter 2 - Literature Review the "summary" of all this discussion results in a problem statement. This problem is focused on the issue of accuracy and predictability of a classifier algorithm (an ANN) used in a critical application when required to classify points outside the boundary of the training set. As such the problem is stated as:

"How to ensure the accuracy and predictability of a classifier, used for head movement classification, for data points outside the boundary of the training set."

3.3 Representative Data & Effective Boundaries – Derivation

In providing a solution to the problem as given in Section 3.2 a helpful start is to take another look at the problem tree and identify some contributing solutions that conceptually address the problem (see below).

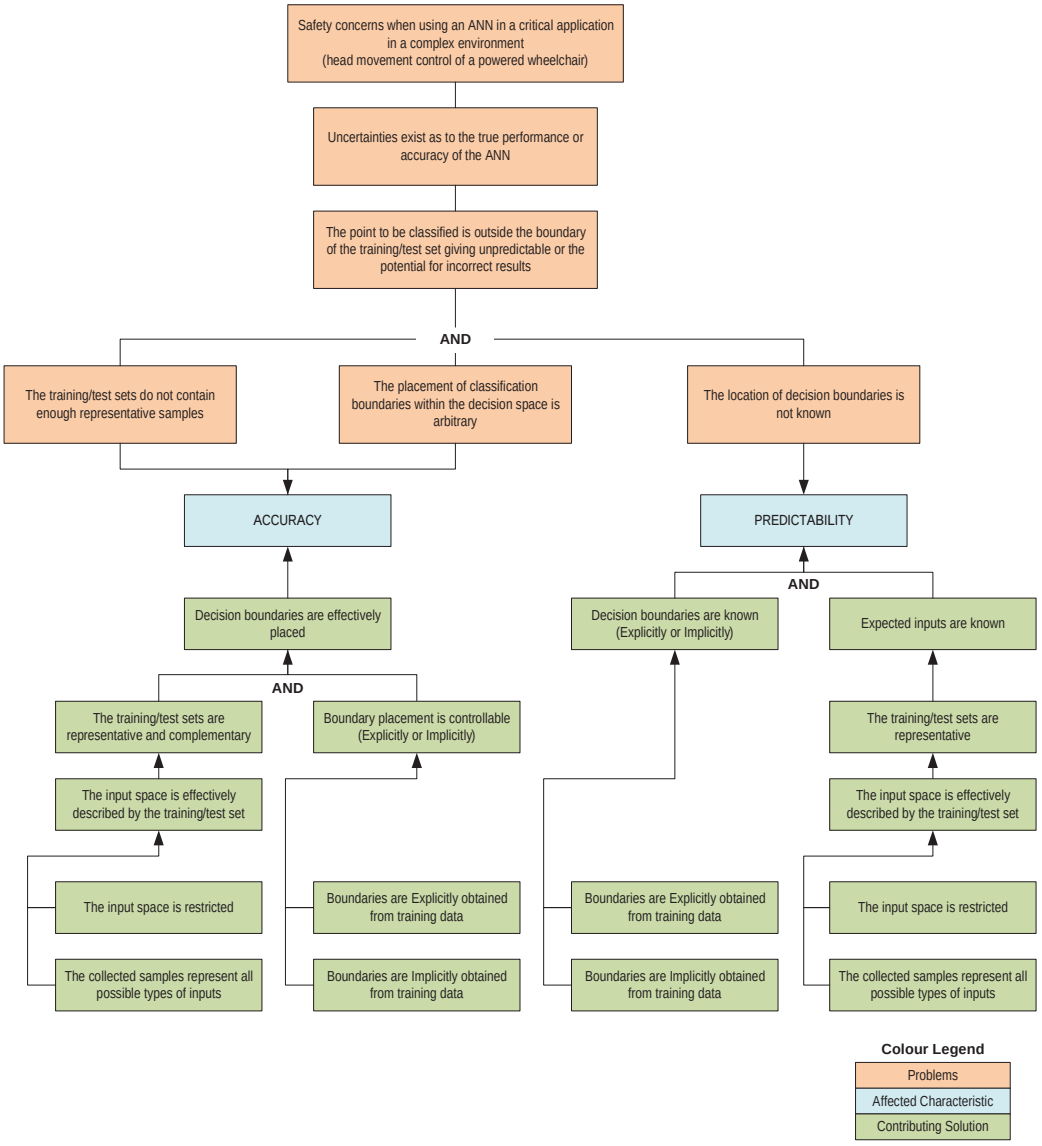

Figure 3-11: Problem/Solution Tree

As mentioned in Section 2.4, the conceptual solution to the problem is to have effectively placed decision boundaries that are known. For this to occur two important factors must be addressed:

- Have a training/test set that is representative and complementary; and
- Use boundaries that are either implicitly or explicitly defined.

The issue of representative data was discussed in Section 3.2.1.1 Representative Data, but to ensure effective boundary placement the representative data must also include all types of data that would be likely to be seen at the input to the classifier. Note, that some of this data would be considered "outside the boundary" of a conventionally trained ANN.

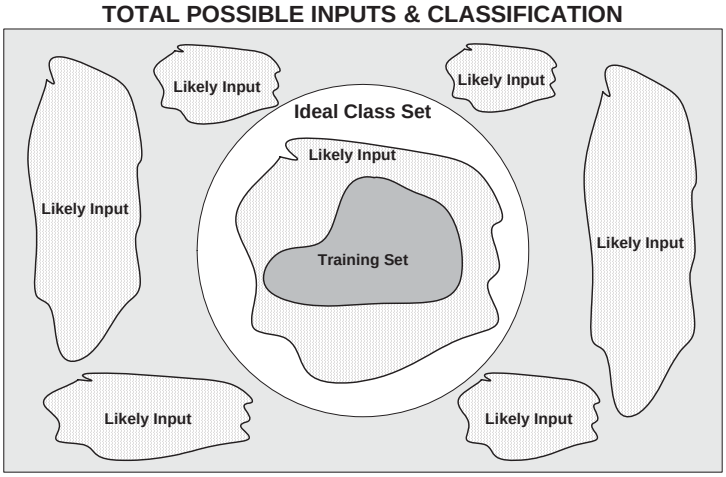

Figure 3-12: Likely Inputs and a Training Set

As can be seen in the figure above, the classifier is likely to see data "outside the boundary" of the training set and also outside the boundary of the "Ideal Class Set". At this stage the classifier is relying on the training set data (and the training itself) to create a boundary that effectively classifies the input data.

Ideally, the result of training would be to create a boundary that encompasses the likely inputs for the "Ideal Class Set", but does not include other types of data that are considered as "Likely Inputs" to the classifier (as shown below).

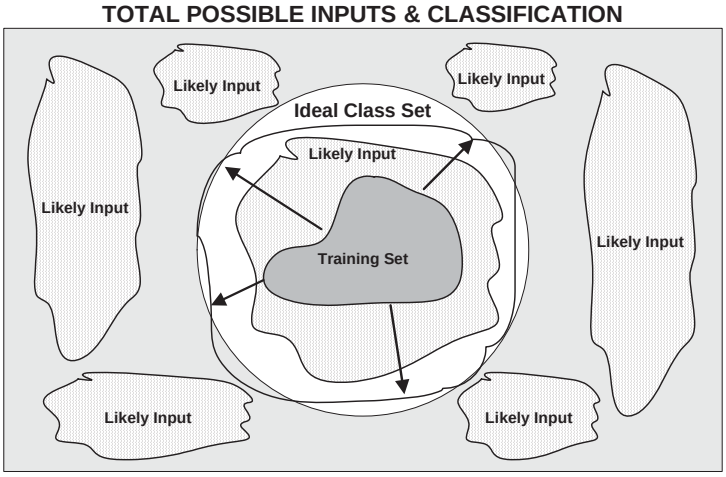

Figure 3-13: Ideal Result of Training

However, this ideal result of training is unlikely, and the more likely result is to "over generalise" or "under generalise" resulting in boundaries that are considered ineffective. As can be seen in the figure below, likely inputs that are not members of the "Ideal Class" to be classified would be incorrectly classified as such.

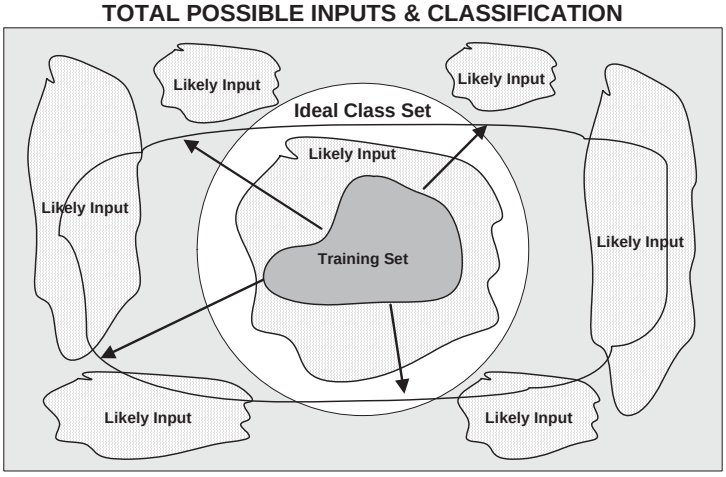

Figure 3-14: Likely Result of Training – Over Generalisation

Also, as seen below in the "under generalisation" case, the likely inputs belonging to the "Ideal Class Set" would not be classified correctly either.

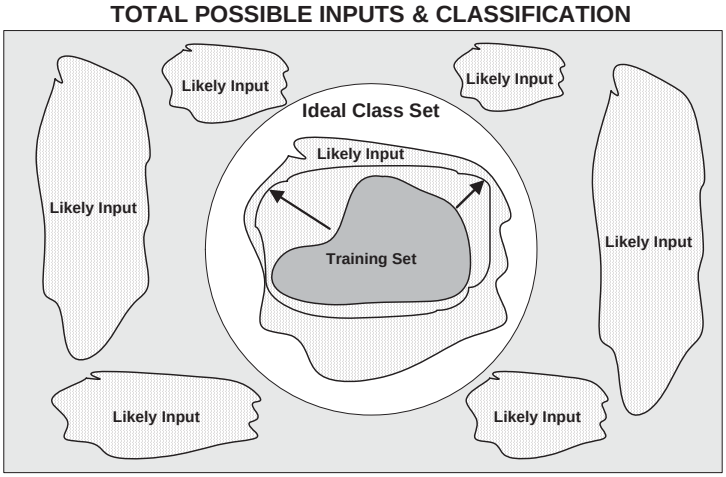

Figure 3-15: Likely Result of Training – Under Generalisation

As seen in figures above (Figure 3-13, Figure 3-14, and Figure 3-15) the training set data provided is not fully representative of the "Likely Inputs" to the classifier and as such impacts the accuracy of the classifier. Also, the formation of the boundaries affected by the training data is not known, which impacts the predictability of the classifier.

To address these shortcomings what is needed is an ability to form effective boundaries that are known from training data.

3.3.1 Creating Effective Boundaries with Training Data

To form a boundary with training data, "positive" samples or class samples must be provided, as these are the patterns to be recognised and "negative" samples or "null" samples must also be provided. The key here is the provision of "negative" or "null" samples, which are complementary to the class set.

Note, while literature doesn't explicitly state this, some literature alludes to this and the understood intention could meet this criteria [6, 7, 52, 62].

These “negative” or “null” samples also exist for multi-class problems, unless the input space would only see these “positive” samples. In this case each class provides the complementary data to be able to form a boundary.

Without the inclusion of complementary training points the ANN would be unable to create adequate boundaries, as shown in the figure below. In this case there are only nine possible input points and only one training point is provided. As can be seen the entire input space is classified as “class 1”.

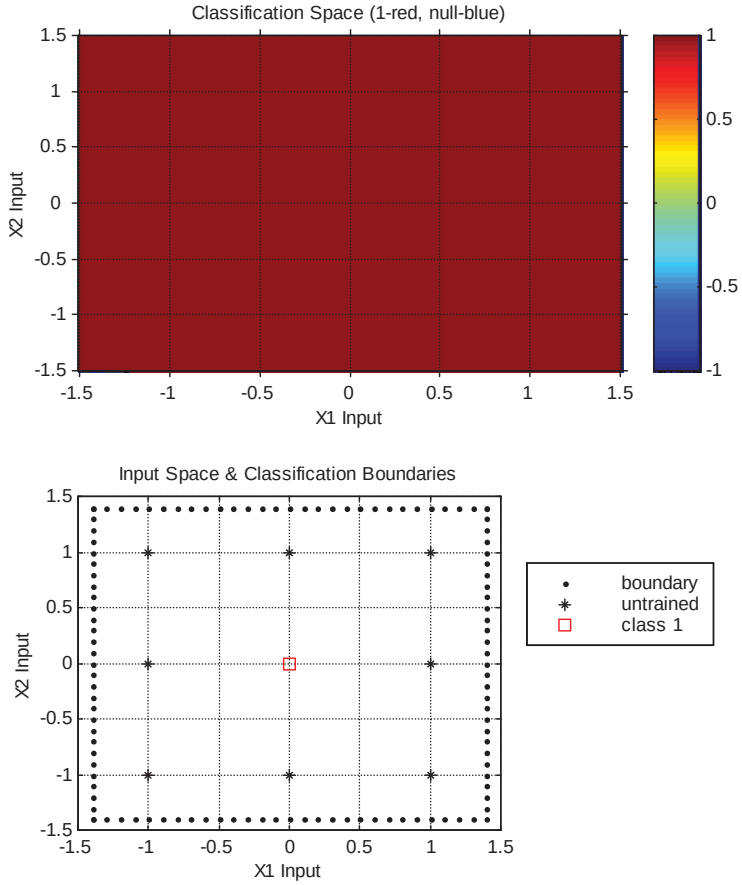

Figure 3-16: NN 2D Plot -1 Class, 1 Training Input

If however we start to add complementary data points to the training set, boundaries start to be created. Again, using the simple nine input example, if for the one class problem we add a "null" point to the training set, there is now a boundary created which divides the input space into "class 1" and "null" possibilities.

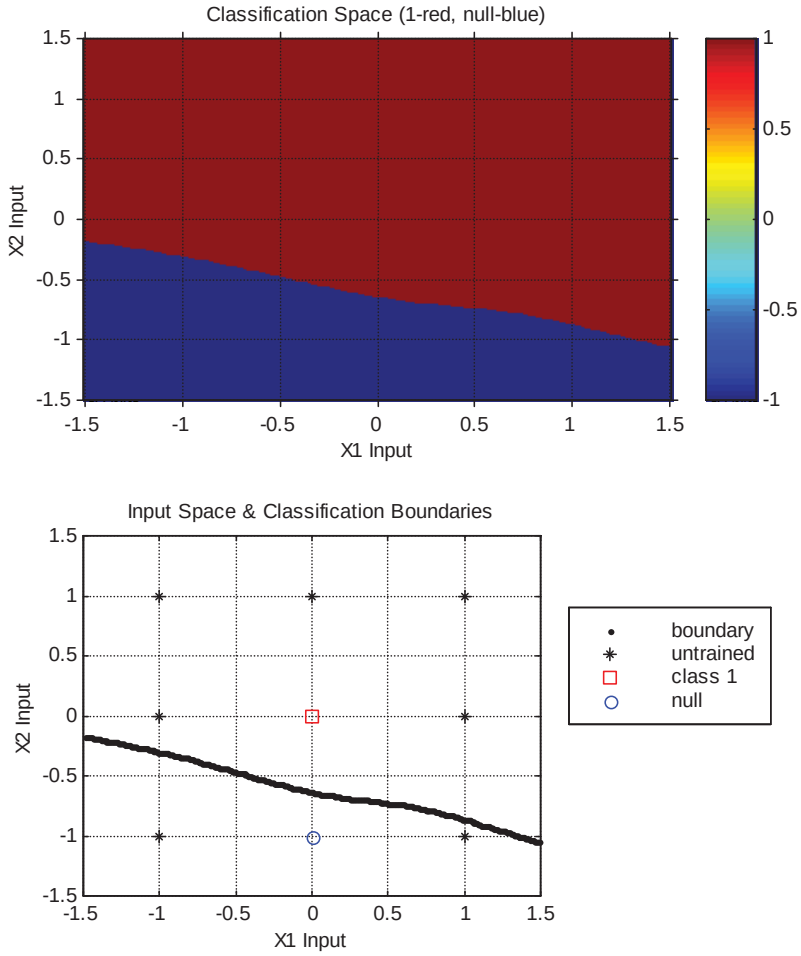

Figure 3-17: NN 2D Plot -1 Class, 1 Positive & 1 Negative Training Input

3.3.1.1 Identifying the True Class Boundary

Without doubt the biggest contributor to uncertainty with an ANN classifier is knowledge of the boundary or the "true class" set for a given problem. Without this information, knowledge regarding the representativeness of any collected training data will not be known.

To know the "true class" boundaries with 100% certainty would mean that knowledge to the class membership of every possible input point is known. For large problems this may not be possible and for some problems the possibility of contradictory class memberships may exist, further compounding uncertainty.

For real problems it would be unlikely to be able to identify the true set memberships of every possible input point or of just a specific class. As this would be the case, strategies to deal with this are required. A suggested strategy includes:

- Identify the class or set memberships, both positive and negative, that could be expected within an input space; and
- For labelling purposes, create rules that help to explain why a point would be considered as part of that class or not of that class; and
- With these rules identify input points or patterns either side of the boundary of a decision or rule. Note that these points need to be considered as being so close to the boundary that it doesn't matter if either point is misclassified (this is the "do not care zone or buffer").

Remembering that by creating rules, boundaries are being created that can only assist in identifying the true "class" boundary, not replace it.

3.3.1.2 Forming Effective & Representative Class Boundaries

Given representative or effective training sets consisting of both "positive" and "negative" examples, ideally generalisation would occur and formation of the boundaries would be close to the ideal class boundary (see figure below).

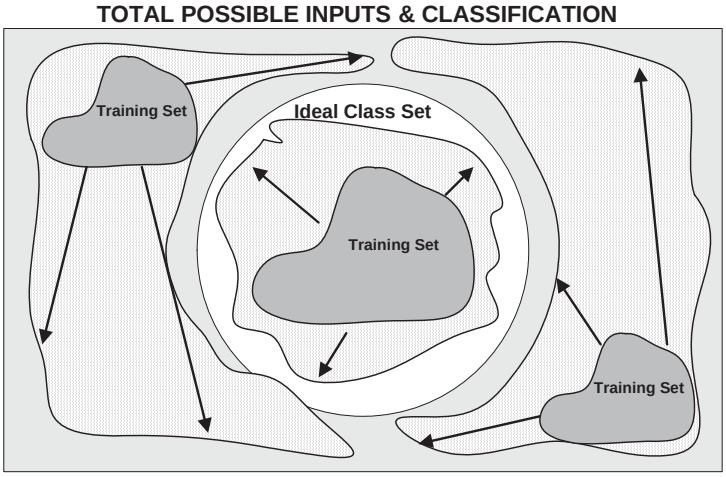

Figure 3-18: Positive and Negative Patterns and Close to Ideal Generalisation

Ideally, once generalisation has occurred there would be a zone around the "ideal class" boundary, which could be considered as a "do not care zone" as it is so close to the ideal boundary, that information regarding correct class membership is not possible to know and therefore could be considered as not important or not relevant.

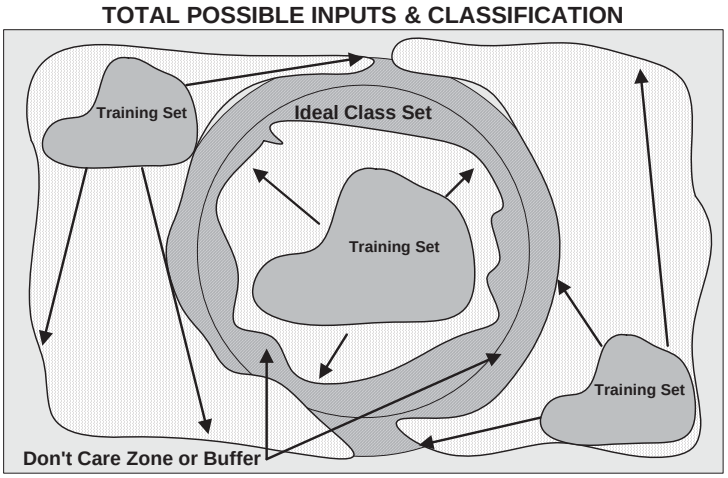

Figure 3-19: The Do not Care Zone or Buffer

Although, this is showing what is an ideal scenario, where the "positive" and "negative" training samples allow for an effective generalisation around the ideal class boundary. Realistically, the location of the ideal class boundary may never be truly known. However, more importantly, this boundary should be able to be effectively created or discovered and placed in the "do not care zone or buffer" by the correct selection of "positive" and "negative" training samples.

This approach of allowing a boundary to form within an area that could be considered as a "do not care zone or buffer" is an implicit approach to effective boundary formation. This implicit approach uses the concept of data "outside the boundary" of the original training set to augment the training set to enable effective implicit boundaries to be formed.

3.3.1.3 Obtaining Effective Training Data

Given that we want to effectively create boundaries with our training data and for generalisation to close in on the "true class" boundaries, how can this effective data be collected? There are two methods that can be employed, one using real data, and the other using artificial data.

3.3.1.3.1 Real Data

The use of real data requires the collection of samples and the creation of patterns from this data. As discussed in section 3.3.1, both "positive" and "negative" samples are required to form a boundary, and both "positive" and "negative" patterns should be created from the collected samples.

Usually, the creation of "positive" samples from collected data doesn't pose much of a problem, however the creation of "negative" samples may. In this case, thorough review of the collected data may be required to understand the types of "negative" patterns that can be created. In doing this more than one type or class of "negative" pattern could be discovered.

To assist in creating or identifying patterns from collected data, rules could be used to help filter the data as discussed in section 3.3.1.1, noting that labels would most probably need to be checked manually.

3.3.1.3.2 Artificial Data

Artificially derived data as discussed previously, could consist of data obtained from a hyper-box as proposed by [6, 7]. In this work a hyper-box was used to form a boundary outside the "positive" training patterns. Once the boundary was established, points on the boundary were collected and added to the training set. These points are the "negative" or "null" training points.

On the face of it, a reasonable proposition and when dealing with simple 2-dimensional examples would be achievable, but may not be effective in more complex situations where input size if potentially large (as is the case with the head movement control).

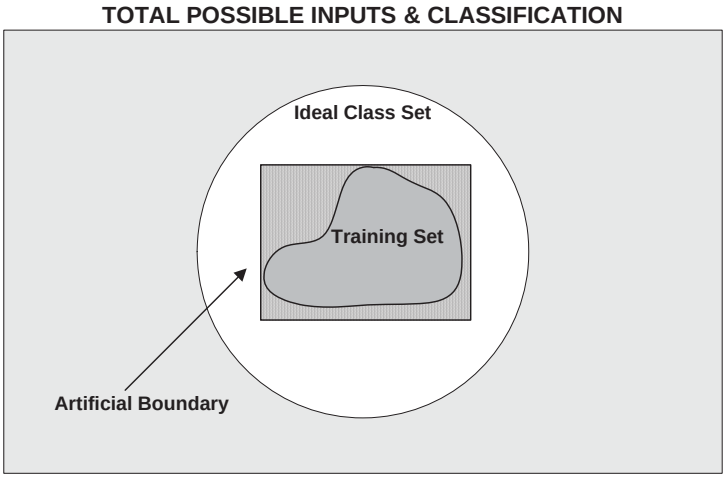

Figure 3-20: Artificially Generated Data – Shortcomings

However, one of the questions resulting is “how much expansion should the box have?” To overcome the shortcomings of the previous work (see Figure 3-20) it is proposed to use the “real data” to assist in optimising the boundary created by the hyper-box (Figure 3-21).

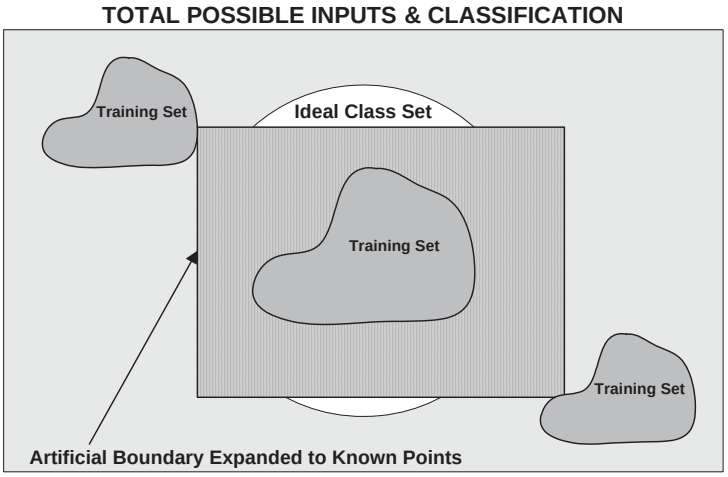

Figure 3-21: Artificially Generated Data with an Optimised Hyper-Box

3.3.2 Explicit and Implicit Boundaries

As discussed, effective boundaries require that the training data be representative of all the “Likely Inputs” to the classifier, not just the class specific data (positive data), but also those inputs that would be considered outside the boundary of the class specific training set or sets for multi-class problems (negative data).

This combination of both “positive” and “negative” representative data becomes a complementary set of training data, which can enable the formation of effective boundaries either implicitly or explicitly.

With an ANN boundary formation is going to be implicit unless all data points, which consist the “Likely Inputs” are used for training. The concept of the implicit boundary relies on the existence of a “do not care zone or buffer” that is not affected by the random boundary formation within that zone (see 3.3.1.2 Forming Effective & Representative Class Boundaries).

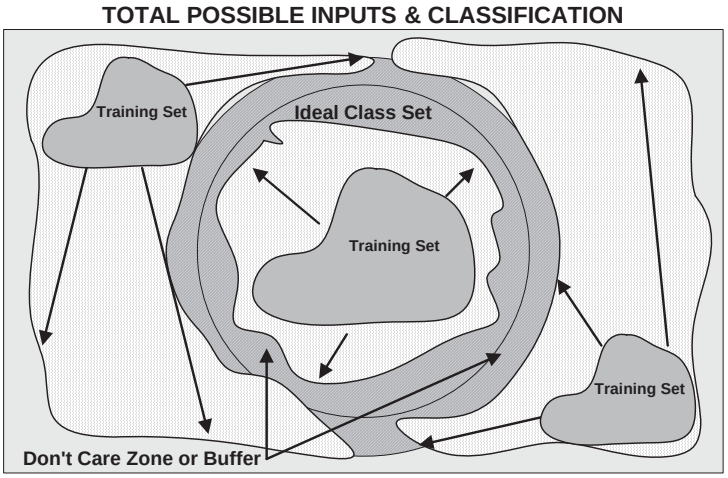

Figure 3-22: Implicit Boundary Formation

This implicit approach to boundary formation while addressing an accuracy issue still has a problem with predictability, as the boundary locations are not known. To address this problem an explicit approach can be used for boundary formation (see below).

In this example the boundary is explicitly defined (shown as a box here) and contains the training set and part of the "Ideal Class Set". By default any data points not within the explicit boundary are excluded.

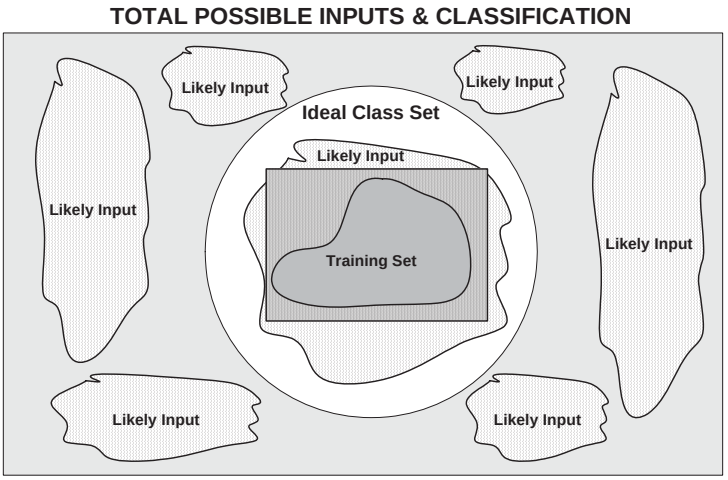

Figure 3-23: Explicit Boundary Formation

As shown above this explicit boundary although including desired points may not be optimal and an expansion of the boundary is required to include additional points that are not in the training set. This additional expansion can be seen as a method of "generalisation" applied to this explicit boundary.

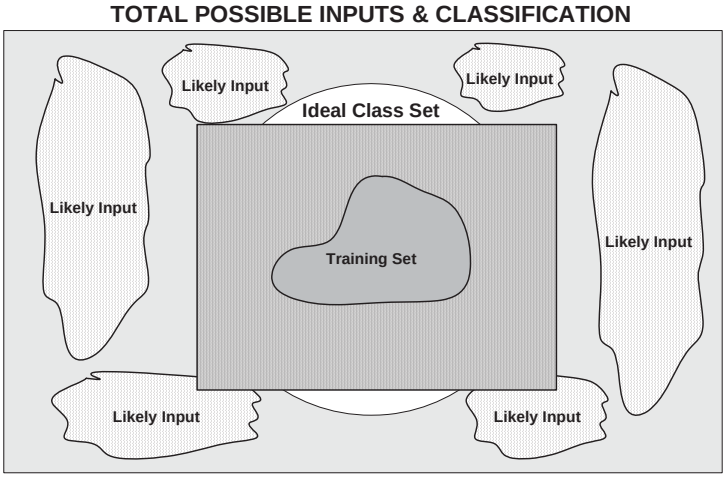

Figure 3-24: Explicit Boundary Formation - Expansion

However, this expansion may not be ideal and could start to include other inputs that are not considered as part of the "Ideal Class Set". In this case the boundary expansion needs to be controlled with additional training data (see below).

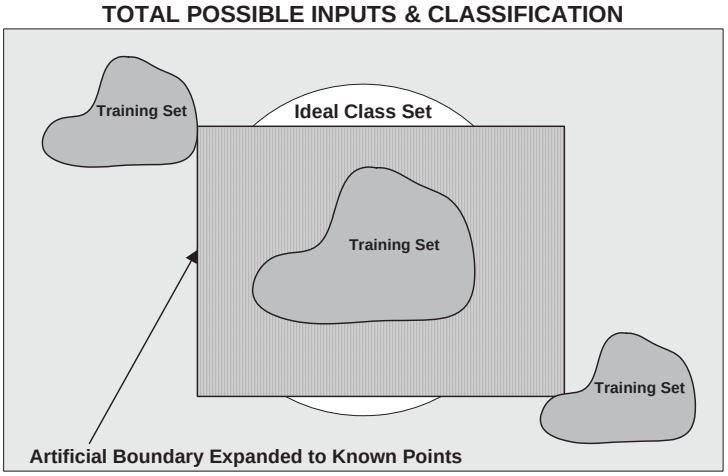

Figure 3-25: Explicit Boundary Formation – Controlled Expansion

Again, this may not be the ideal or optimal situation and further refinement could be required. A potential refinement is to allow the explicit boundary to expand to known points, but to include regions that are explicitly defined, but are not included in the class set covered by the expansion (as shown below).

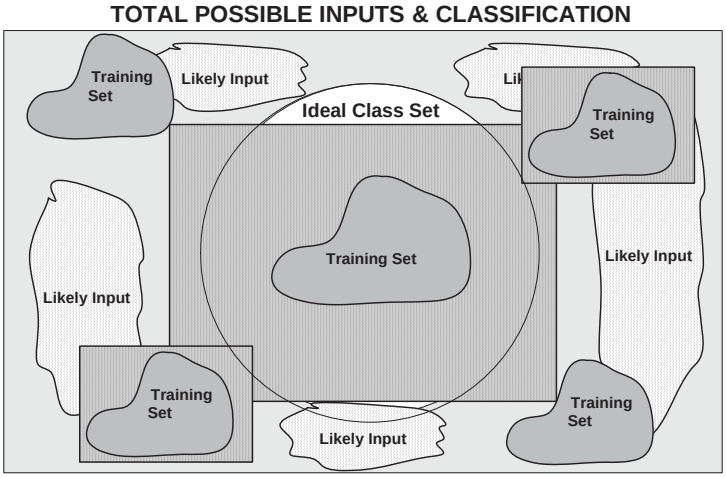

Figure 3-26: Explicit Boundary Formation – Optimised

This approach could be considered as an optimal way to use explicitly defined regions and boundaries to classify data.

3.4 Discussion and Conclusion

To address the stated problem, which concerns the accuracy and predictability of an ANN used in a critical application, two conceptual solutions were discussed:

- Training/test data must be representative and complementary; and
- Boundaries are implicitly or explicitly defined.

These conceptual solutions become merged when the outcome is really "effective boundaries", which address the issues of accuracy and predictability.

The important concept here is "effective boundary formation", which itself requires that the training/test data be representative and complementary of the likely inputs to the classifier and it is this data that allows the classifier to form either implicit or explicit boundaries.

To achieve an implicit boundary in an ANN the idea of comprehensive "complementary" training/test data was discussed (see Section 3.3.2), which requires that the training data also contain those points likely to be seen by the classifier, but outside the boundary of the original training set. This allows the classifier to form effective implicit boundaries, which enable the ANN to address the issues of accuracy and predictability.

However, it must be noted that predictability cannot be fully addressed with an ANN as the boundaries are randomly generated, but if the boundary is created in the "do not care zone or buffer" it becomes less of an issue.

The use of explicit boundaries such as a hyper-box (for example) immediately gives predictability as the boundary is known. However, the location of boundaries may not be truly effective and requires "expansion" outside the boundary of the training data (a form of generalisation) to achieve improved performance.

This expansion can be controlled by the use of data outside the boundary of the training set, which is provided by having comprehensive "complementary training/test data. Further adjustments and optimisations can be made which involve that use of additional regions and boundaries based on the training data to achieve improved performance.

The combination of explicitly defined boundaries that are optimised with additional regions and boundaries provides a conceptual solution that addresses both accuracy and predictability.

Chapter 4 - Bootstrap and ROC Analysis for Head Movement

4.1 Introduction

This chapter describes the common methodology used for the experiments in the subsequent chapters. All the experiments undertaken in the subsequent chapters use the same data set with training patterns extracted to create various types of training sets to test different classifier algorithms.

In testing the different classifiers two major methods were combined to create what is being termed "The 0.632+ Bootstrap ROC Analysis". This combination provided a method to create 0.632+ Bootstrap sets in which an ROC Analysis had been conducted.

From these sets statistical parameters for mean, upper, and lower confidence intervals could be obtained for each threshold value of sensitivity and specificity. This data in turn is used to compare each algorithm.

4.2 Testing Classifier Algorithms

A lot of literature has been written about this topic and previously the methods of choice were related to cross-validation methods. Using some of the general suggested methods from [63] and [64], a real application is used to make comparisons and a form of cross-validation (0.632+ Bootstrap Method) is also used to account for variance and bias. Further, statistically valid comparisons are made to enable any conclusions drawn to be meaningful.

Of major influence in understanding how to make valid comparisons with classification algorithms is the work undertaken by [59] which clearly highlights the influences of various factors such as the algorithm, training set, test set and the interactions between all. This work particularly was important in gaining an appreciation of the influences of variance in any comparison and it led to the use of the 0.632+ Bootstrap Method [65].

4.2.1 ROC Analysis

The influence of medical research is also important as the evaluation of diagnostic testing introduces the two major measures of sensitivity and specificity [56] and [57]. From these two measures the receiver operating characteristics curve (ROC curve) can

be evaluated and is a particularly useful measure for evaluating classifiers used in medical diagnostics [44].

Extensive literature exists regarding the use of ROC curves and the application to classifier algorithms especially in imaging and medical diagnosis, such as [45], [46] and [47] for example. This is being taken up by the pattern analysis and machine learning community and generally the ROC curve area is recommended as the measure of choice [48].

The method being used for the experiments is the ROC area under the curve, which is calculated from the sensitivity and specificity measures. Also from the ROC curve the threshold value that provides the best sensitivity and specificity value for subsequent comparison is obtained. For an example of the application of bootstrap to ROC area under the curve see [49] and [50] (note that these examples do not provide final sensitivity and specificity measures for each threshold level, which enables construction of ROC curves).

4.2.2 The 0.632+ Bootstrap Method

The use of pattern recognition algorithms for classification purposes and inferences made regarding the accuracy of trained algorithms to populations outside of the training set is considered problematic. Recent statistical methodologies from [65] consider these problems by using modern statistical methods, such as “the bootstrap” and “randomisation”.

In addressing the problems of “error rate” or “accuracy” of algorithms, the effects of small sample sizes, the effects of the training set and the test set are taken into account. All of these factors generate “bias” and [65] attempts to deal with these issues by generating a bias correction. This corrections is performed using the “0.632 Bootstrap Method” and an improvement called the “0.632+ Bootstrap Method” is employed to generate a final value that has accounted for possible bias introduced by the test set and by random chance.

The final method used in [65] is known as the “0.632+ Bootstrap Method” and is recommended as the method of choice when attempting to determine the accuracy or

error rate of a given algorithm type [59]. This becomes especially useful when comparing algorithms.

4.2.3 The 0.632+ Bootstrap ROC Analysis

This term "0.632+ Bootstrap ROC Analysis" is a combination of the 0.632+ Bootstrap and of ROC Analysis. As described in the previous two sections, ROC is considered a measurement of choice in comparing two algorithms, and the Bootstrap helps alleviate the effects of small sample sizes, the effects of the training set and test set.

It is quite a reasonable approach to combine the two for the purposes of assessing the performance of each head movement classification algorithm.

Also, as discussed in the previous section others made mention of Bootstrap and ROC Analysis (see [49] and [50]), however their approach is to create a single measure such as Area Under The Curve (AUC) for the ROC analysis then create Bootstraps of this measure to obtain a mean and corresponding confidence intervals.

For the experiments presented here it is proposed to create Bootstrap sets that contain the sensitivity and specificity at each threshold value then obtain a final set of values containing the mean and corresponding confidence intervals for sensitivity and specificity. From these measures of sensitivity and specificity the ROC curves and Area Under the Curve can be generated.

Once all the ROC data is obtained comparisons of the different algorithms can be made.

4.3 Classifier Types Under Test

In this thesis, we are mainly concerned with the performance of one major classifier type the Artificial Neural Network. The Artificial Neural Network was considered as it is the classifier type given in [2], [3], and [4] for the selected critical application.

The work shown in [7] provides an interesting insight into how the decision boundary placement may be influenced by the use of data points outside the class boundary derived from a hyper-cube and hence the decision to attempt to improve the classifier performance through the use of such hyper-geometric boundaries.

Of the possible hyper-geometric boundaries, such as the hyper-sphere, hyper-cube, hyper-box, hyper-rectangle, the hyper-rectangle is used, which builds upon suggestions made in [7], where a hyper-cube was used as a method to define a boundary outside of the training set data for the creation of additional training data.

Note, that all the stated types of hyper-geometric boundaries (sphere, box, cube, rectangle) are essentially equivalent in implementation in that a cluster is created, a centre is found, and a boundary is created around the edge of the data.

Note, that for brevity the detailed workings of Artificial Neural Networks and k-means clustering will not be addressed in this thesis. For further information consult any reference texts on Artificial Neural Networks, statistical pattern recognition, and/or artificial intelligence (references used for this thesis were [8], [54], [66], [67] and [60]).

4.3.1 Artificial Neural Networks

The Artificial Neural Network (ANN) used here is a Multi-layer Feed Forward Perceptron Neural Network. The detailed workings of this "common" style of classifier can be found in any reference text on the subject (see [8] or similar texts for explanation of the workings of Artificial Neural Networks).

The architecture of the ANN used here is shown below.

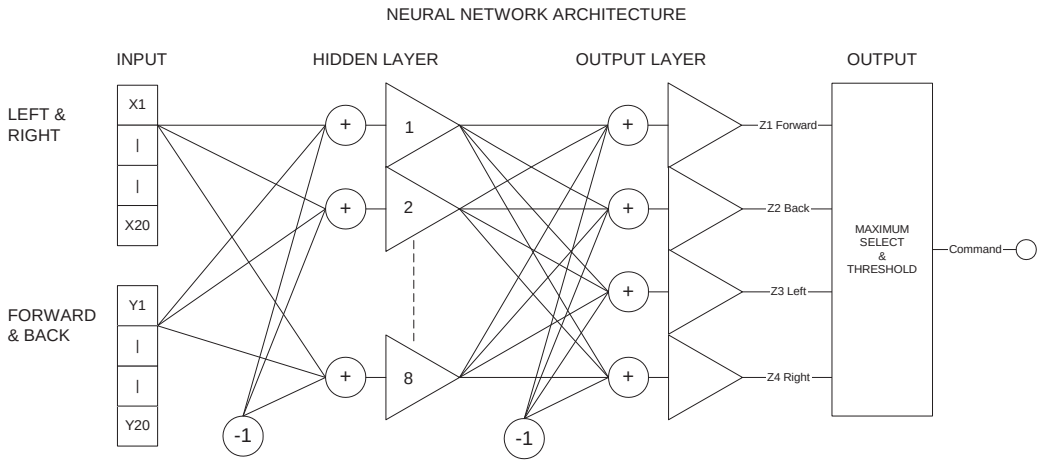

Figure 4-1: Neural Network Architecture

4.3.2 Hyper-Rectangle Basis Function

A variation on the hyper-cube [6, 7, 37, 39], that gives more flexibility is the "Hyper-Rectangle basis function", which uses training data to obtain clusters (k-means clustering [60]) for each class and then obtains the minimum and maximum values for each data input dimension that define the cluster to form a Hyper-Rectangle.

Each cluster becomes a node in the network with classification consisting of a comparison for each dimension value with the minimum and maximum for that dimension. If each dimension is within the minimum and maximum value, for a particular class a classification is made. Where classifications overlap the class with the largest absolute average and maximum value is identified and assigned to that class.

Obtaining an optimal network is as simple as using training set error to control the number of clusters and the amount of generalisation or expansion of the Hyper-Rectangles (while maintaining aspect ratio). Performance is then measured with the test set to obtain a measure of sensitivity and specificity.

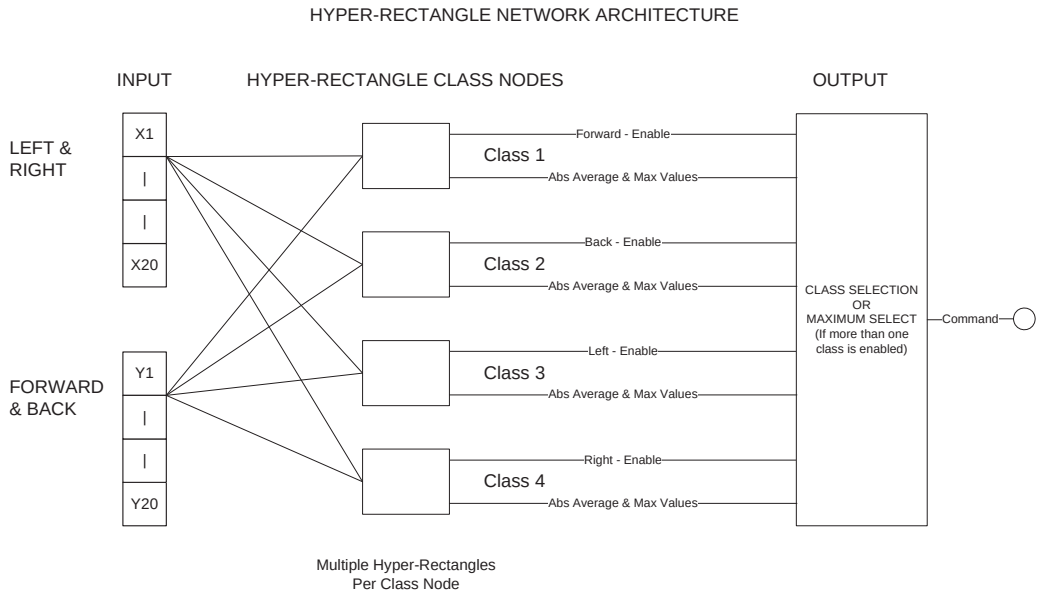

Figure 4-2: Hyper-Rectangle Network Architecture

An examination of the Hyper-Rectangle in two-dimensions shows that there is a region created by the maximum and minimum values obtained from training data for each of the 40 dimensions. If a pattern is contained within one of the regions it is classified as such. Two examples are shown below, one that is an artificial representation and the other from real data.

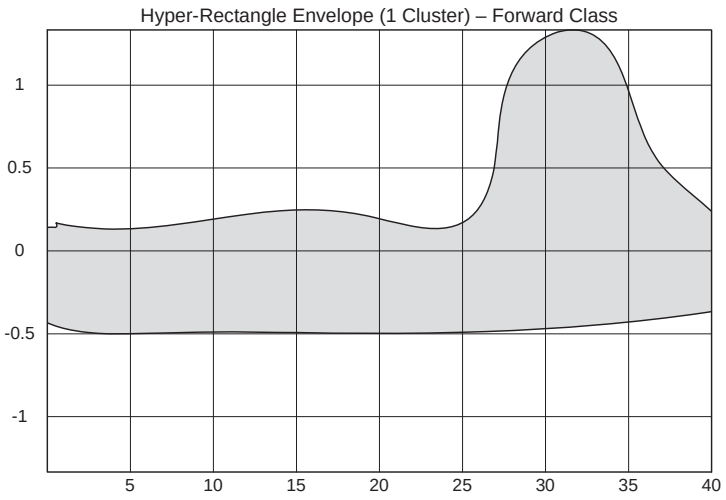

Figure 4-3: Hyper-Rectangle Envelope – Forward (Artificial)

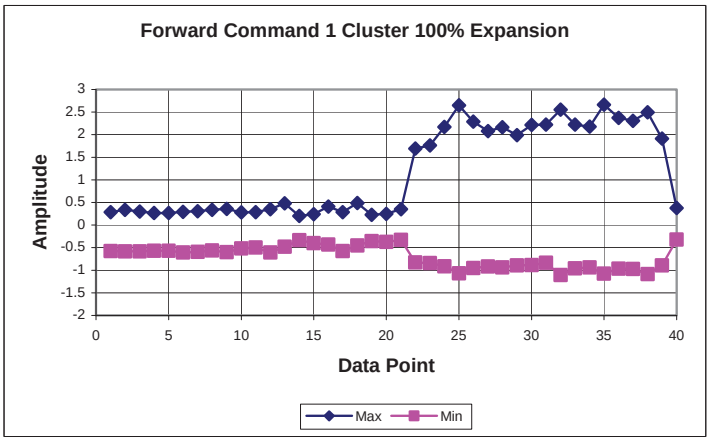

Figure 4-4: Hyper-Rectangle Envelope – Forward (Real Data)

The figure below shows a typical Hyper-Rectangle region for one cluster for a Backward Command. Two examples are shown below, one that is an artificial representation and the other from real data.

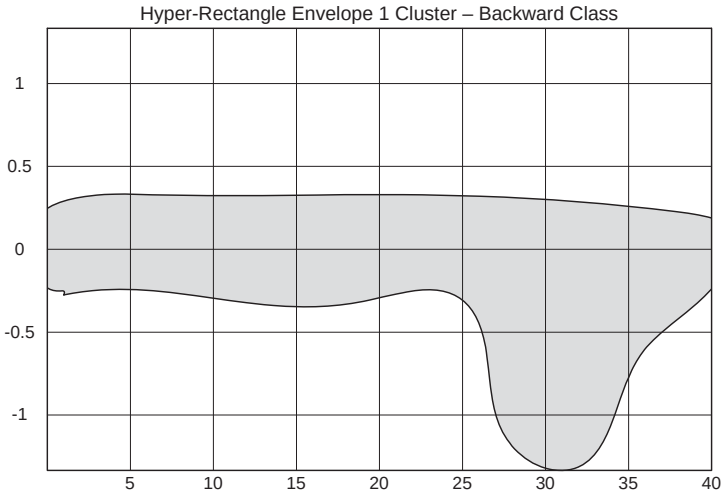

Figure 4-5: Hyper-Rectangle Envelope – Backward (Artificial)

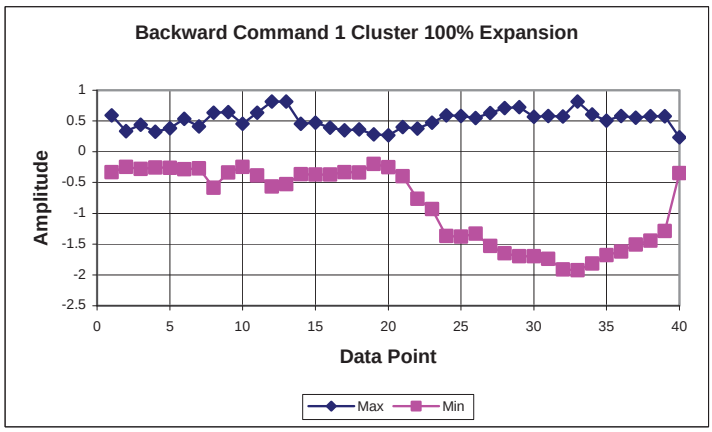

Figure 4-6: Hyper-Rectangle Envelope – Backward (Real Data)

The figure below shows a typical Hyper-Rectangle region for one cluster for a Left Command. Two examples are shown below, one that is an artificial representation and the other from real data.

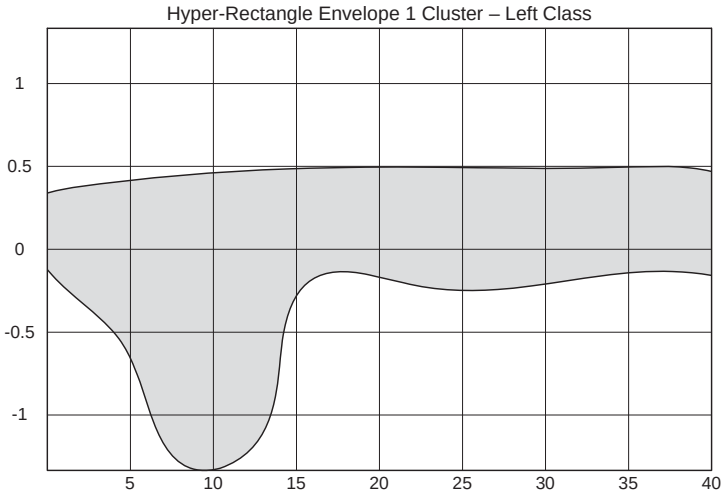

Figure 4-7: Hyper-Rectangle Envelope – Left (Artificial)

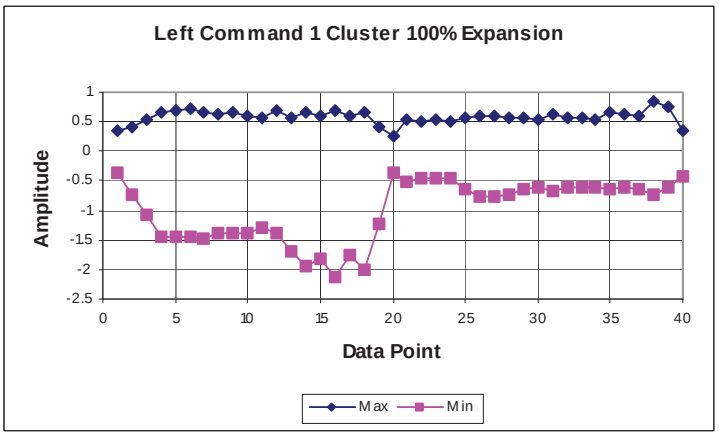

Figure 4-8: Hyper-Rectangle Envelope – Left (Real Data)

The figure below shows a typical Hyper-Rectangle region for one cluster for a Right Command. Two examples are shown below, one that is an artificial representation and the other from real data.

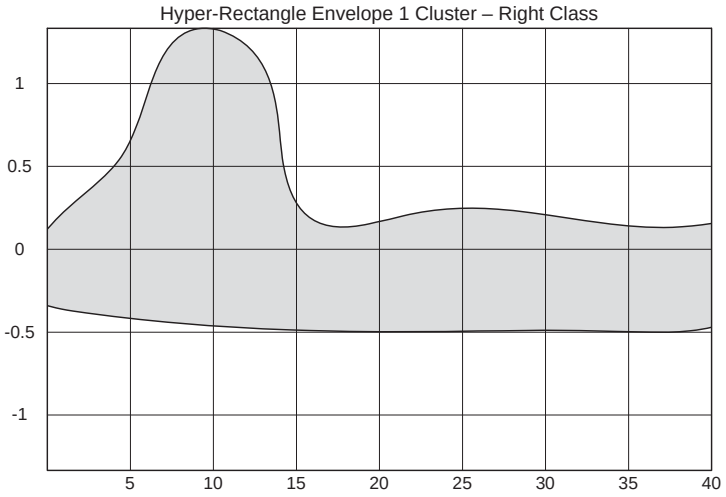

Figure 4-9: Hyper-Rectangle Envelope – Right (Artificial)

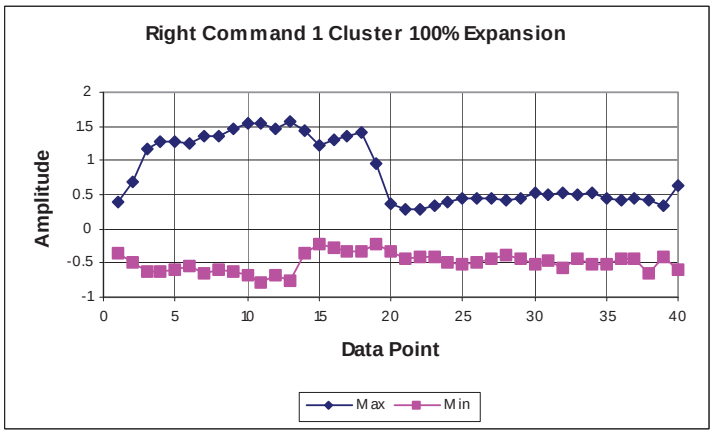

Figure 4-10: Hyper-Rectangle Envelope – Right (Real Data)

The figure below shows all classes at 100% expansion past the originally extracted envelope position (one cluster per class).

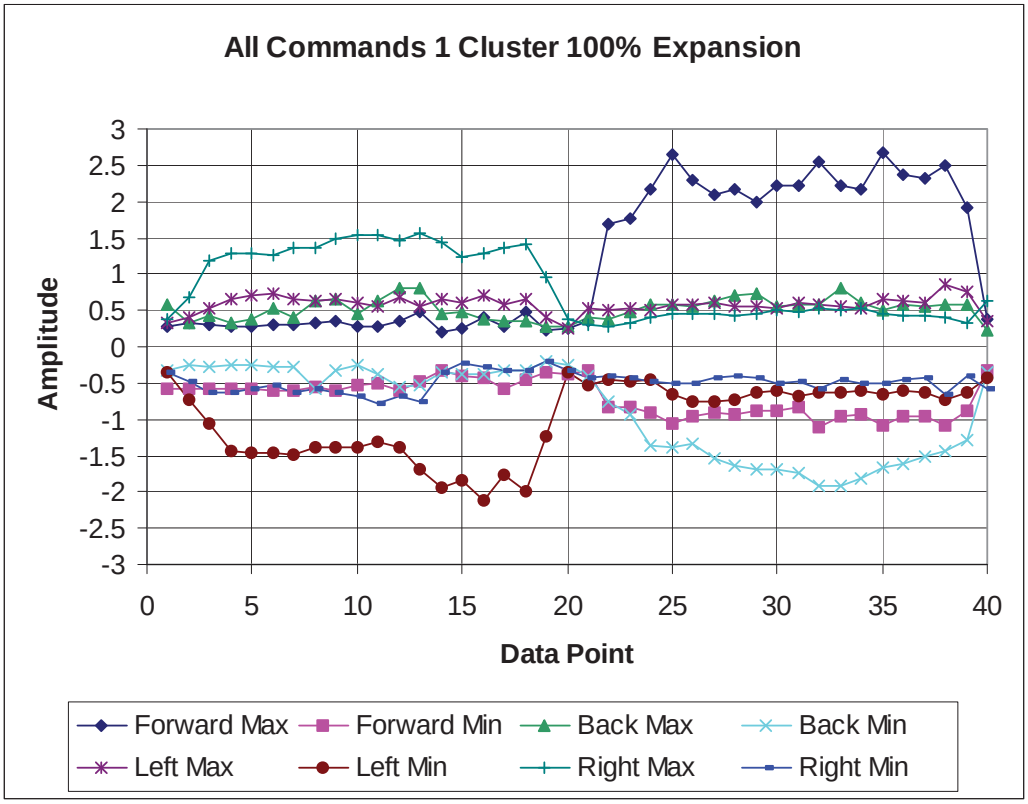

Figure 4-11: Hyper-Rectangle Envelope – All Classes (Real Data)

The figure below shows the classification boundary for the forward command using three clusters. The selection of the optimal number of clusters is based on error obtained from sensitivity and specificity measures.

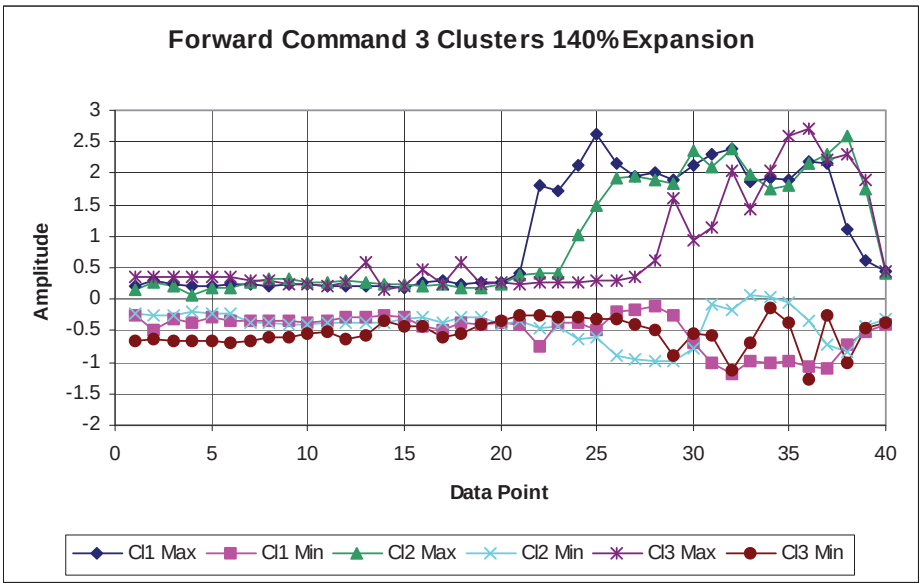

Figure 4-12: Hyper-Rectangle Envelope Multiple Clusters – Forward (Real Data)

4.3.3 A Region and Boundary Classifier

A Region and Boundary Classifier is based on the Hyper-Rectangle Basis Function for the classification of each class (see Section 4.3.2). However, it incorporates additional regions created from the training data that capture other types of non-command signals that are subsequently incorporated into class boundaries when expanded to an optimal size (see Section 3.3.2). These additional regions are called the Horizontal Null Region and the Vertical Null Region. The architecture of the Region and Boundary Classifier is shown below.

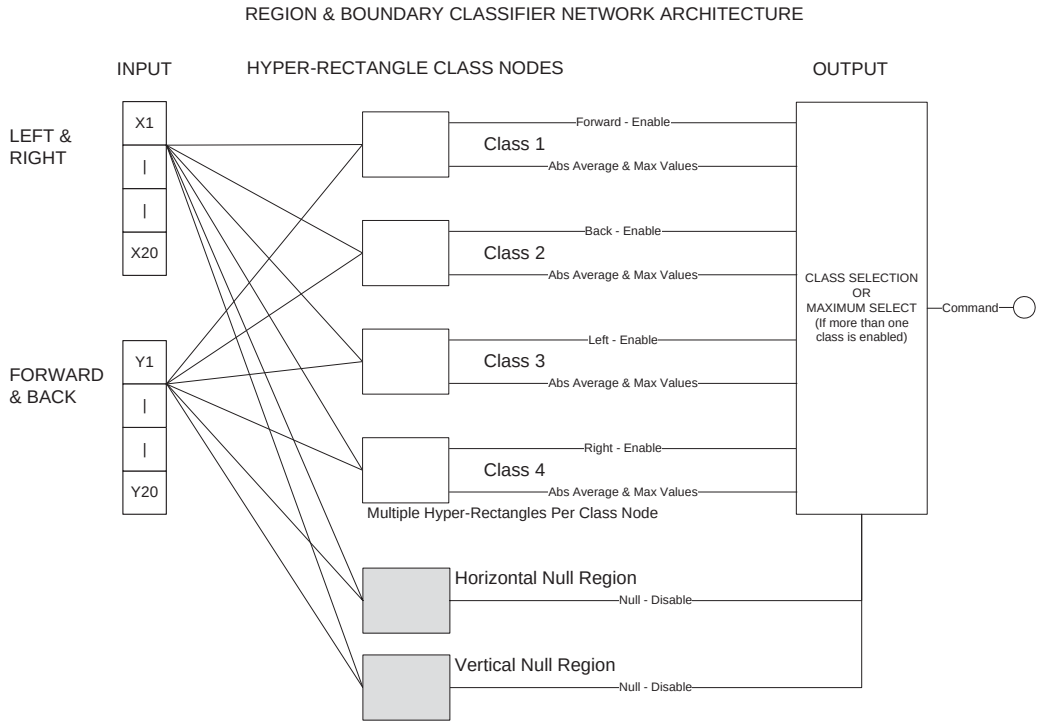

Figure 4-13: Region and Boundary Classifier Network Architecture

The Horizontal Null Region is an area obtained from the training data that creates a maximum and minimum region to capture those patterns that may be included in the class Hyper-Rectangles, but are not part of that class set. Two examples are shown below, one that is an artificial representation and the other from real data.

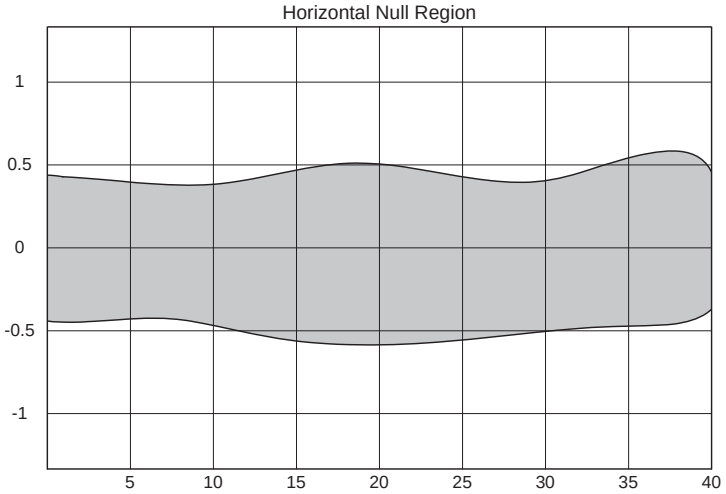

Figure 4-14: Region and Boundary Classifier Horizontal Null Region - Artificial

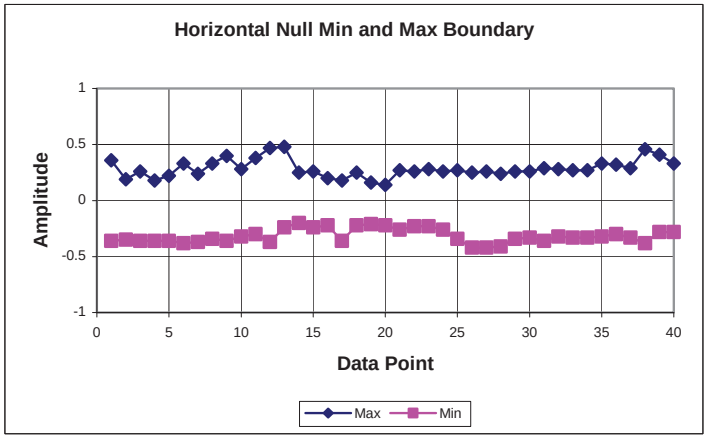

Figure 4-15: Region and Boundary Classifier Horizontal Null Region - Real

The figure below shows the Horizontal Null Boundary implemented with each class for one cluster. Note, how the Null Region captures the common space between each class to ensure patterns such as a "Null" or "Flat Line" are not picked up by one of the classes.

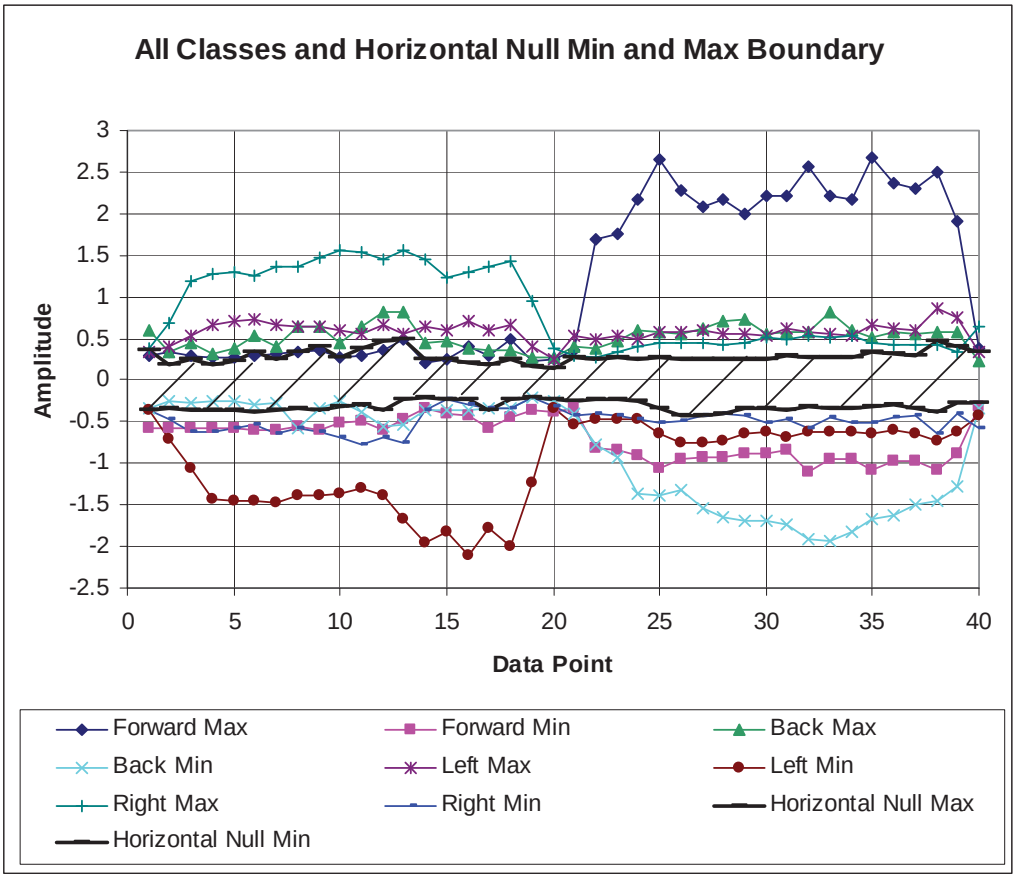

Figure 4-16: Region and Boundary Classifier Horizontal Null Region - Real

The Vertical Null Region are areas at the beginning and end of the x-axis and y-axis parts of the pattern data (points 1, 20, 21, and 40) that captures those patterns that may be included in the class Hyper-Rectangles, but are not part of that class set. This type of region ensures that a command starts and finishes within the two-second (or 20 data points) window. An example is shown below.

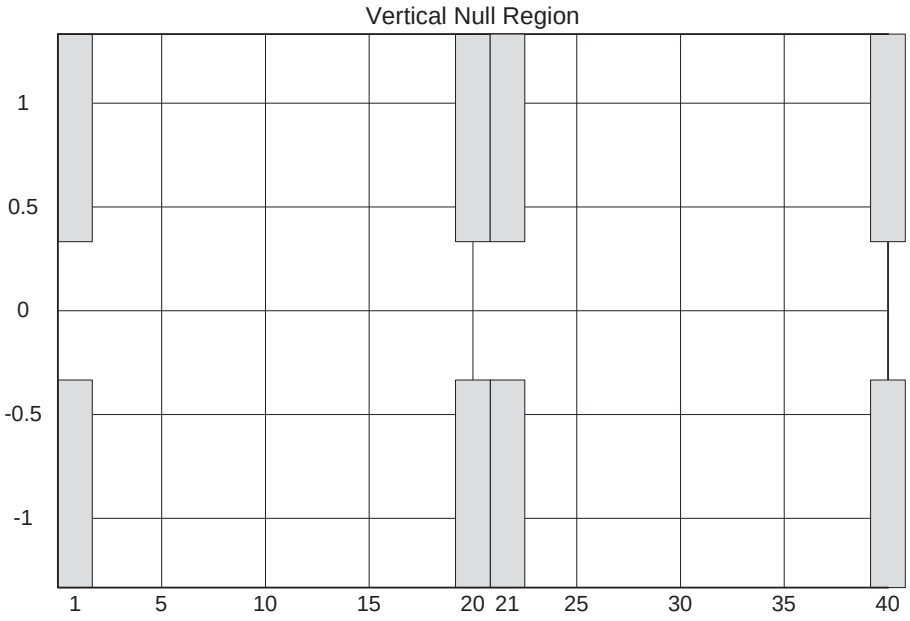

Figure 4-17: Region and Boundary Classifier Vertical Null Region

4.3.4 Definitions

To assist in understanding, some definitions need to be provided to explain the terminology being used. The definitions below explain what is meant by the terms and jargon used in the context of head movement wheelchair control:

- Real-Time Data Signal: Refers to a two second window of discrete samples from a two-axis accelerometer. This window of 20 X-axis and 20 Y-axis data samples over 2 seconds is used as input to the classifier to interpret head movements;

- Input Space: Refers to the set of total possible inputs that could be presented as an input to the classifier. For example, if the classifier has two inputs or two dimensions and each dimension can take on three possibilities (-1, 0, 1) then the input space would consist of all possible combinations of these values for two

dimensions [(-1, -1), (-1, 0), (-1, 1), (0, -1), (0, 0), (0, 1), (1, -1), (1, 0), (1, 1)], for a total number of $3^2 = 9$, possible inputs;

- Output State: Given a real time data signal this refers to the final output of the classifier and the five possible states, forward, backward, left, right, or none of these (undecided or null). In our case, the classifier has four outputs, one for each command class, where the maximum value of the four is selected and compared against a threshold, if the value is above the threshold then the output of the classifier is that class and considered a command, otherwise we have an "undecided or null" state and is considered a non-command or unknown;

- Command: A head movement such as a forward, backward, left, or right nod that is issued by the user either intentionally or unintentionally. This head movement would be considered valid from the user's perspective if information regarding the user intention is unknown;

- Intentional Command: The user has issued a deliberate or intentional command;

- Unintentional Command: The user has issued an unintentional command, either accidentally or due to other circumstances such as environmental conditions;

- Command Signal: A real time data signal representing a command;

- Non-Command Signal: A real time data signal that is not a command;

- Class: Association or grouping of like entities. In this context command signals that belong to either the set of forward, backward, left and right head movements. Each one is considered a class, e.g. the forward class etc. Note, that both command and non-command signals could be considered a class, but in this context we take only take action on a command signal;

- Pattern Class: Refers to either the forward, backward, left or right classes either individually or collectively;

- Command Class: A super set comprising of the forward, backward, left, and right classes;

- Non-Command Class: A super set comprising the complement of the Command Class i.e. those signals that do not belong to the Command Class;
- True Class: Another name for Command Class;
- Positive Class: Another name for Command Class;
- False Class: Another name for the Non-Command Class;
- Negative Class: Another name for the Non-Command Class;
- Undecided: When used in the context of a final output classification, undecided is that which isn't, forward, backward, left or right. This state consists of the set of unknown data and the known non-command data;
- Nothing: Another term for undecided;
- Null: Another term for undecided;
- Known: Refers to a real time data signal where information regarding which class it belongs to is "known". Loosely defined, training and test data is "known". Examples of this are training and test data containing command signals and non-command signals;
- Unknown: Refers to a real time data signal where information regarding which class it belongs to is "unknown". Note, that an unknown real time data signal could belong to either the command or non-command signal sets;
- Outside the boundary: Refers to a real time data signal that is not part of the pattern class training data and is therefore "unknown". Another way to term this is that the signal is outside the boundary of what is known. This definition also implies that there is a boundary created by the training data;
- Correctly Classify: Refers to the classifier correctly classifying an input with the resulting output for any known or unknown data. This makes the assumption that all unknown data can be separated into the two super sets of commands and non-commands and further separated into the four head movement command sets. Note,

that in making this definition we are implying a possibility of knowledge existing as to the classification of the input space;

4.4 Data Collection

4.4.1 Overview

In conducting the experimentation the original real-time data was obtained from able and disabled users and from this data the bootstrap training and test sets were created. For each bootstrap training set, each type of classifier algorithm was created, trained, and subsequently tested with the corresponding bootstrap test set.

Using the 0.632+ Estimator sensitivity and specificity was calculated for each bootstrap set and from these measures the receiver operating characteristic curve (ROC) was calculated. Finally, using the measures of sensitivity, specificity, and ROC comparisons were made.

4.4.2 Data Source

The real-time head movement data used came from a pool of previously collected and labelled data used within the UTS wheelchair control research group. The method of collection is detailed in reference [4].

Briefly, the data was obtained from ten adults aged between 19 and 56, which was previously collected with approval from the UTS Human Research Ethics Committee and informed consent from the volunteers. Of the ten volunteers, three had high-level spinal cord injuries (C4 and C5) and were unable to use a standard joystick to control a wheelchair. The remaining seven volunteers were considered able-bodied and did not have any conditions that affected their movements. Table 4-1 below shows the number of original head movement patterns collected from each category of volunteer (able or disabled). Also, the numbers of derived non-command patterns are shown in Table 4-2 as well.

Table 4-1: Originally Collected and Labelled Head Movement Data - Command

Volunteer Category	Head Movement Type			
	Forward	Backward	Left	Right
Able	79	83	93	96
Disabled	23	34	9	7
TOTAL	102	117	102	103

Table 4-2: Originally Collected Head Movement Data – Non-Command

		Non-Command Pattern Types	
Number	Acronym	Description	Patterns Extracted
1	Yp	Y-axis above 0.5 threshold, X-axis is a flat line	12577
2	Yn	Y-axis below -0.5 threshold, X-axis is a flat line	14172
3	Xp	X-axis above 0.5 threshold, Y-axis is a flat line	11998
4	Xn	X-axis below -0.5 threshold, Y-axis is a flat line	12215
5	XpYp	X-axis above 0.5 threshold, Y-axis above 0.5 threshold	227
6	XpYn	X-axis above 0.5 threshold, Y-axis below -0.5 threshold	97
7	XnYn	X-axis below -0.5 threshold, Y-axis below -0.5 threshold	109
8	XnYp	X-axis below -0.5 threshold, Y-axis above 0.5 threshold	316
9	FXp	Y-axis has valid forward nod, X-axis above 0.5 threshold	0
10	FXn	Y-axis has valid forward nod, X-axis below -0.5 threshold	0
11	BXp	Y-axis has valid backward nod, X-axis above 0.5 threshold	0
12	BXn	Y-axis has valid backward nod, X-axis below -0.5 threshold	0
13	LYp	X-axis has valid left nod, Y-axis above 0.5 threshold	0
14	LYn	X-axis has valid left nod, Y-axis below -0.5 threshold	0
15	RYp	X-axis has valid right nod, Y-axis above 0.5 threshold	0
16	RYn	X-axis has valid right nod, Y-axis below -0.5 threshold	0
17	Null	Flat Line	75837
		TOTAL	**127548**

Note: these patterns were extracted from the collected real-time data signal files using a sliding window and a combination of manual labelling and a simplified rule set to assist in identifying each type of non-command.

4.4.3 System Signal Flow

The real-time data signal for wheelchair control uses the system described in [2] and is shown below

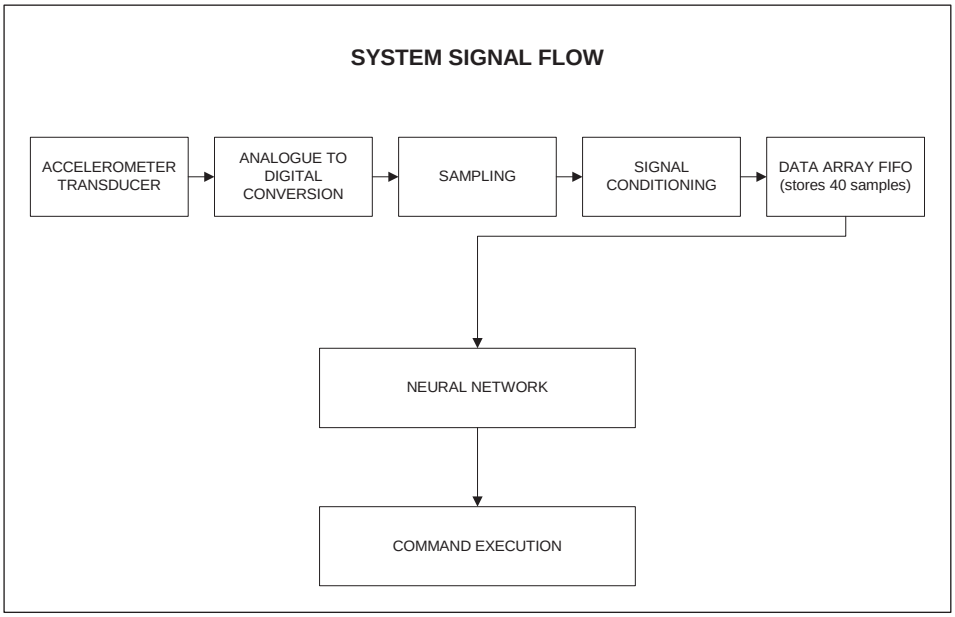

Figure 4-18: System Signal Flow

An analogue transducer (two-axis accelerometer) provides the head movement sensing data with head tilt forward, back, left, and right. This data is sampled through an eight-bit analogue to digital converter and either used in the control of the wheelchair or stored for later processing.

An example of a real time signal is shown below. As can be seen there are peaks representing a head movement, either forward, back, left or right, this is called command data. Also, notice that areas not part of a command head movement, but rather the constant real time stream of data is present, this data is called non-command data.

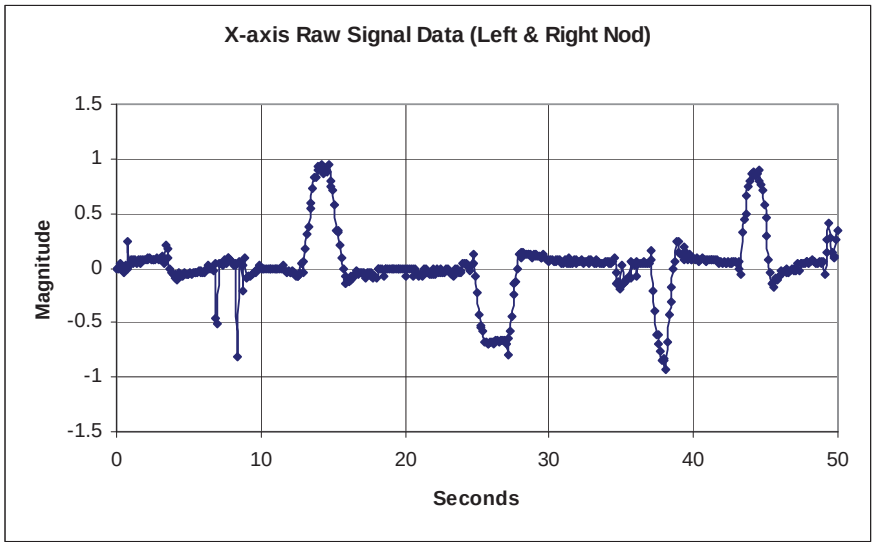

Figure 4-19: X-axis Accelerometer Real Time Data Signal

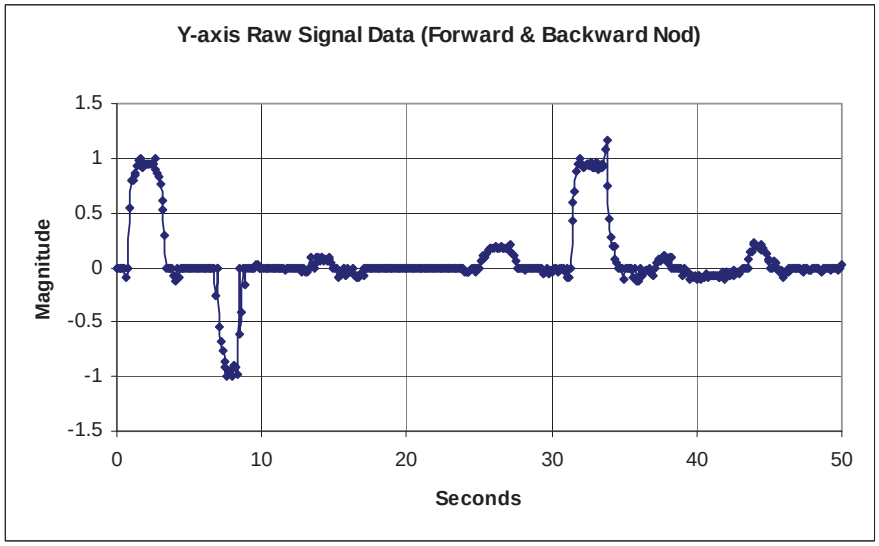

Figure 4-20: Y-axis Accelerometer Real Time Data Signal

4.5 Pattern Set Creation

From the real-time head movement data collected, pattern sets were created using a combination of manual labelling and a sliding window with a simplified rule application process to assist labelling, and labelled as either forward, backward, left, or right commands or a non-command. A command consists of either a forward, backward, left or right head movement and a non-command consisted of all those data that do not belong to the set of commands.

Note that of these non-commands seventeen types or classes of signal were identified from the enormous amount of data obtained from the sliding window.

The creation of these patterns consisted of the following:

1. For each sample type create a window of data points that is the size of the input pattern to be classified. In this case 20 data points from an X-axis and 20 points from a Y-axis accelerometer was used (approximately a 2 second sample, where sampling time is 0.1 seconds). Note that the X-axis and Y-axis signals are effectively sampled at the same point in time.
2. Slide the two-second-sample window (containing 20 X-axis and 20 Y-axis samples) across the entire sampled data set to create patterns that would be seen in real time by the classification algorithm. Noting, that once a pattern is labelled a command the sliding window skips ahead two seconds to ensure that the discovered pattern is not reused.
3. To assist in labelling the patterns each X-axis and Y-axis pattern was passed through a set of rules to provide a first pass unambiguous classification that labelled the patterns as forward, backward, left, right (command) or a non-command (positive, negative, or null). The rules were based on the following:

- A valid head movement is approximately one second in duration therefore a valid command head movement should be completed easily within two seconds or 20 samples.

- At the beginning and end of a head movement the magnitude of the signal should be below a threshold of 0.3 in magnitude at the window sample points of 1 and 20.

- A valid head movement should have at least five continuous data points above a threshold of 0.5 in magnitude.

- A signal that isn't a valid head movement and has at least one data point above a threshold of 0.5 is considered as either an X-positive (Xp) or Y-positive (Yp).

- A signal that isn't a valid head movement and has at least one data point below a threshold of -0.5 is considered as either an X-negative (Xn) or Y-negative (Yn).

- A signal that isn't a valid head movement and has at least one data point above a threshold of 0.5 for the X-axis or Y-axis and at least one data point below a threshold of –0.5 for the other axis is considered as either an "X-positive (Xp) Y-positive (Yp)" (XpYp), "X-positive (Xp) Y-negative (Yn)" (XpYn), "X-negative (Xn) Y-positive (Yp)" (XnYp), or "X-negative (Xn) Y-negative (Yn)" (XnYn).

- A signal that is considered a valid command in one axis only, but the other axis is not a "flat line" or "null" is also considered as a "non-command". For each possible command in one axis, forward, backward, left, or right, the other axis can be considered as X-positive (Xp), X-negative (Xn), Y-positive (Yp), or Y-negative (Yn). This results in combinations of Forward or Backward, X-positive or X-negative (F-Xp, F-Xn, B-Xp, B-Xn), and Left or Right, Y-positive or Y-negative (L-Yp, L-Yn, R-Yp, R-Yn). Identification of these combinations is shown in Table 4-4.

- A signal that isn't a valid head movement and is not an Xp, Xn, Yp, Yn, XpYp, XpYn, XnYp, XnYn, F-Xp, F-Xn, B-Xp, B-Xn, L-Yp, L-Yn, R-Yp, or R-Yn signal is considered a "null" or basically flat line signal.

4. Once all pattern data is labelled by the rule set, merge the corresponding X-axis and Y-axis data patterns into patterns that are 40 samples in length and each pattern is relabelled accordingly. Again, note that for the purposes of this analysis, for a valid command to exist only the X-axis (left or right) or Y-axis (forward or backward) can be present. If both an X-axis and Y-axis command is present, then it is not considered a valid command and is considered as another type of non-command (refer to Table 4-4 for a list of all identified types of "non-command" patterns).

5. Each pattern (consisting of the merged X-axis and Y-axis patterns) and classification was manually checked to ensure the patterns had the correct labelling. If a classification was not correct the label was changed.
6. Finally, the patterns were trimmed to create equal sized command and non-command data sets by trimming the valid command data patterns for each type of head movement to 90 patterns (randomly selected). This created a total valid command set of 360 patterns and then 360 valid non-command patterns were randomly selected to create the two equal sized sets.

Note that not all possibilities of non-command data were available from collected data.

The final number of patterns to be used in the bootstrap sets for command and non-command data shown below in Table 4-3 and Table 4-4.

Table 4-3: Final Number of Patterns Used

Pattern Type				
Forward	Backward	Left	Right	Non-Command
90	90	90	90	360

Table 4-4: Non-Command Pattern Types Used

		Non-Command Pattern Types	
Number	Acronym	Description	Patterns Used
1	Yp	Y-axis above 0.5 threshold, X-axis is a flat line	40
2	Yn	Y-axis below -0.5 threshold, X-axis is a flat line	40
3	Xp	X-axis above 0.5 threshold, Y-axis is a flat line	40
4	Xn	X-axis below -0.5 threshold, Y-axis is a flat line	40
5	XpYp	X-axis above 0.5 threshold, Y-axis above 0.5 threshold	40
6	XpYn	X-axis above 0.5 threshold, Y-axis below -0.5 threshold	40
7	XnYn	X-axis below -0.5 threshold, Y-axis below -0.5 threshold	40
8	XnYp	X-axis below -0.5 threshold, Y-axis above 0.5 threshold	40
9	FXp	Y-axis has valid forward nod, X-axis above 0.5 threshold	0
10	FXn	Y-axis has valid forward nod, X-axis below -0.5 threshold	0
11	BXp	Y-axis has valid backward nod, X-axis above 0.5 threshold	0
12	BXn	Y-axis has valid backward nod, X-axis below -0.5 threshold	0
13	LYp	X-axis has valid left nod, Y-axis above 0.5 threshold	0
14	LYn	X-axis has valid left nod, Y-axis below -0.5 threshold	0
15	RYp	X-axis has valid right nod, Y-axis above 0.5 threshold	0
16	RYn	X-axis has valid right nod, Y-axis below -0.5 threshold	0
17	Null	Flat Line	40
		TOTAL	**360**

4.5.1 Manual Viewing and Labelling Tool

As mentioned in the previous section a manual labelling approach was required to firstly observe the data then subsequently label. Two utilities were created to enable the viewing, labelling, and subsequent checking of all labelled patterns.

The figure below shows a screen shot of one of the tools created to enable a viewing of a collected data file, which can subsequently be labelled or relabelled. Note, that the two vertical red lines show the sample window for each X-axis and Y-axis pattern.

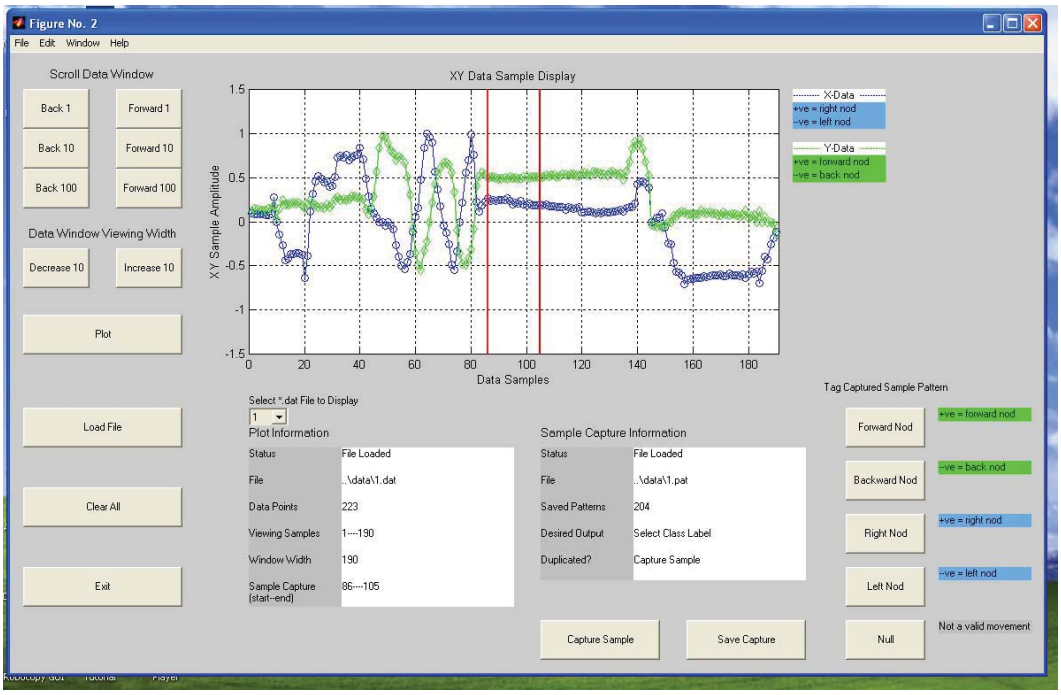

Figure 4-21: Manual Labelling Tool – 190 Samples Wide View

The figure below shows the same data (as above) between data points 86 – 105, but in a closer or "zoomed in" view.

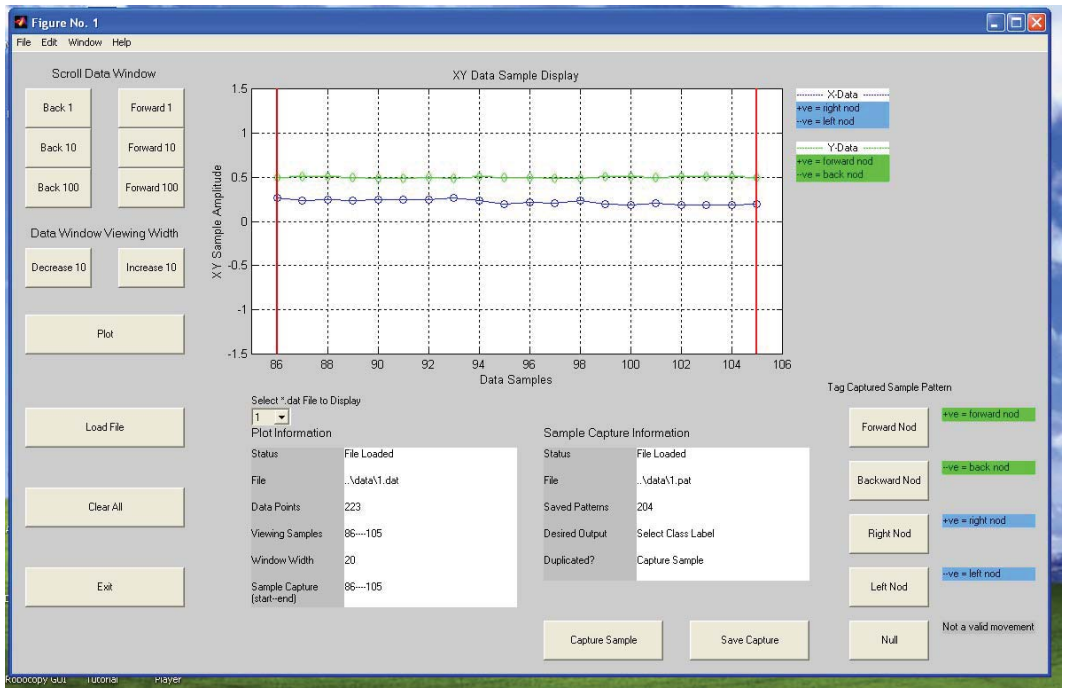

Figure 4-22: Manual Labelling Tool – 20 Samples Wide View

4.5.2 Example Patterns

Examples of merged and labelled patterns are shown in the figures below for both valid commands and non-commands.

4.5.2.1 Command Signals

The following figure shows the four valid nod commands (forward, backward, left, and right).

Note that the sequence of commands as shown (left to right then top to bottom) is Right, Forward, Left, and Backward nods.

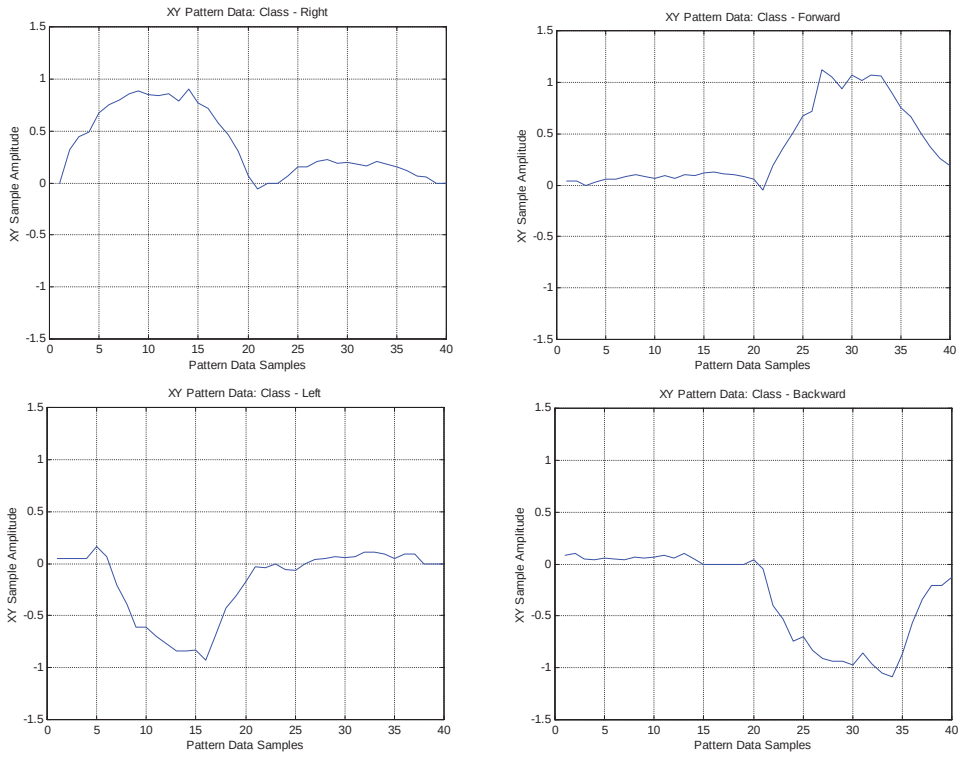

Figure 4-23: Valid Command Training Patterns – R, F, L, B

4.5.2.2 Non-Command Signals

The following figure shows a flat line or "null" signal. This pattern consists of the X-axis component signal (data points 1 to 20) having amplitude between thresholds of –0.5 and 0.5 and the Y-axis component signal (data points 21 to 40) also having amplitude between thresholds of –0.5 and 0.5.

Note: examples of this pattern were found in the collected pattern data.

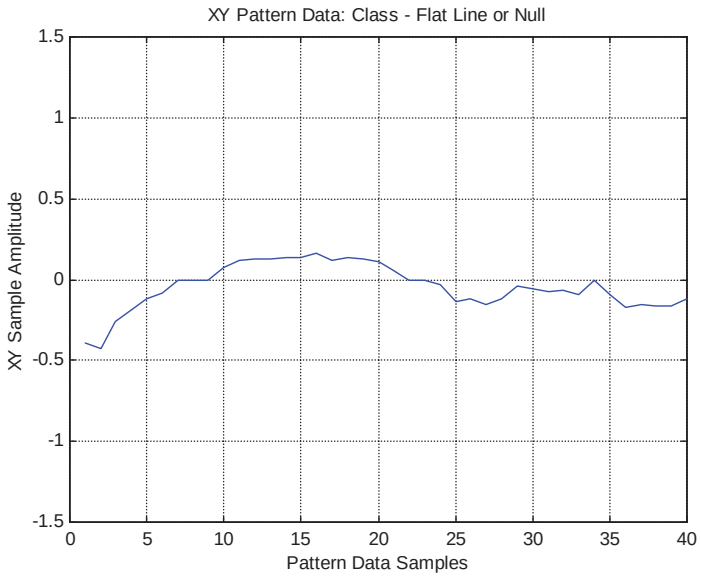

Figure 4-24: Non-Command Training Pattern – Flat Line or Null

The following figure shows “Xp, Yp, Xn, and Yn” non-command signals (examples of these signals were found in the collected training data).

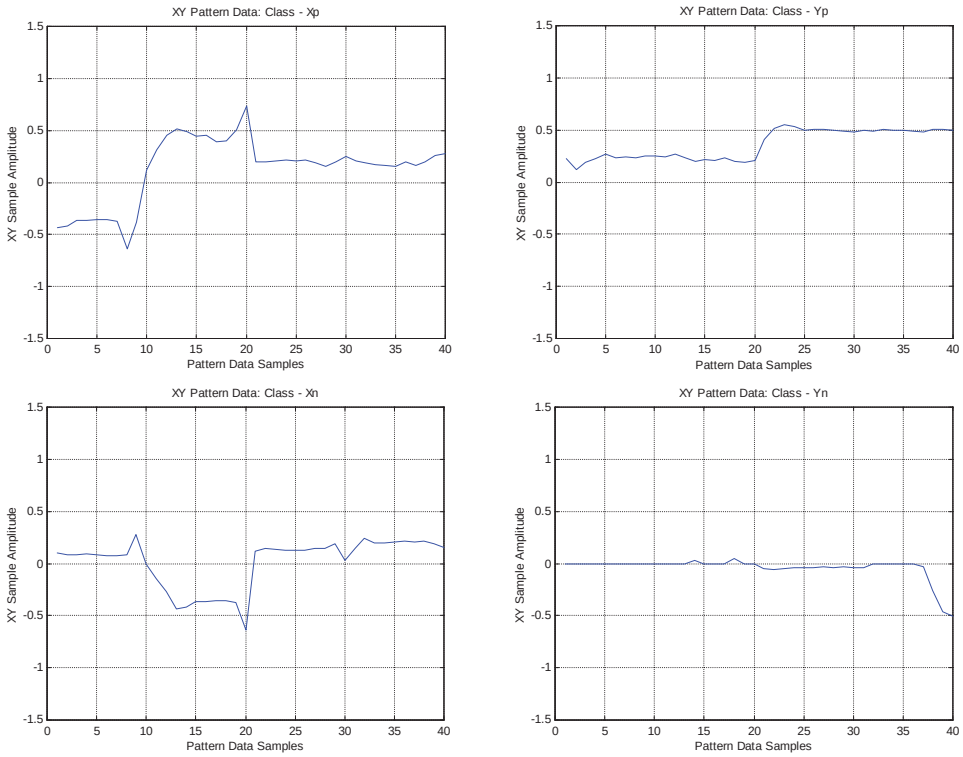

Figure 4-25: Non-Command Training Patterns – Xp, Yp, Xn, Yn

The “Xp” pattern consists of the X-axis component signal (data points 1 to 20) having amplitude greater than a threshold of 0.5 and the Y-axis component signal (data points 21 to 40) having a flat line or “null” signal.

The “Yp” pattern consists of the X-axis component signal (data points 1 to 20) having a flat line or “null” signal and the Y-axis component signal (data points 21 to 40) having amplitude greater than a threshold of 0.5.

The “Xn” pattern consists of the X-axis component signal (data points 1 to 20) having amplitude less than a threshold of -0.5 and the Y-axis component signal (data points 21 to 40) having a flat line or “null” signal.

The “Yn” pattern consists of the X-axis component signal (data points 1 to 20) having a flat line or “null” signal and the Y-axis component signal (data points 21 to 40) having amplitude less than a threshold of -0.5.

The following figure shows “XpYp, XpYn, XnYp, and XnYn” non-command signals (examples of these signals were found in the collected training data).

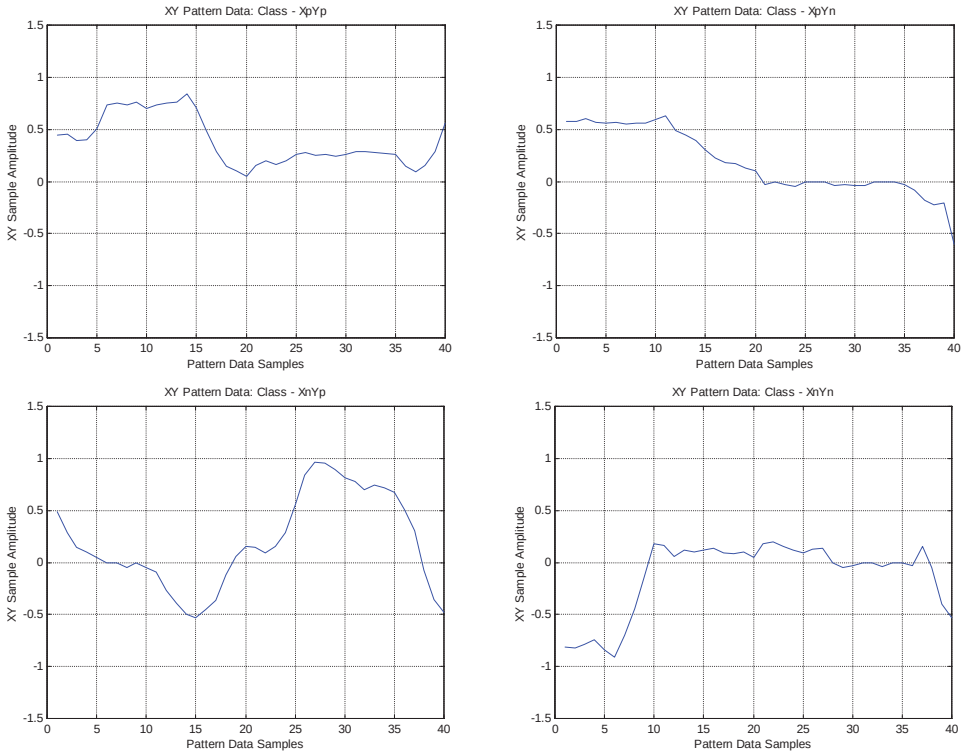

Figure 4-26: Non-Command Training Patterns – XpYp, XpYn, XnYp, XnYn

The “XpYp” pattern consists of the X-axis component signal (data points 1 to 20) having amplitude greater than 0.5 and the Y-axis component signal (data points 21 to 40) having amplitude greater than a threshold of 0.5.

The “XpYn” pattern consists of the X-axis component signal (data points 1 to 20) having amplitude greater than 0.5 and the Y-axis component signal (data points 21 to 40) having amplitude less than a threshold of -0.5.

The “XnYp” pattern consists of the X-axis component signal (data points 1 to 20) having amplitude less than -0.5 and the Y-axis component signal (data points 21 to 40) having amplitude greater than a threshold of 0.5.

The “XnYn” pattern consists of the X-axis component signal (data points 1 to 20) having amplitude less than -0.5 and the Y-axis component signal (data points 21 to 40) having amplitude less than a threshold of -0.5.

The following figure shows “FXp, FXn, BXp, and BXn” non-command signals (no examples of these signals were found in the collected training data).

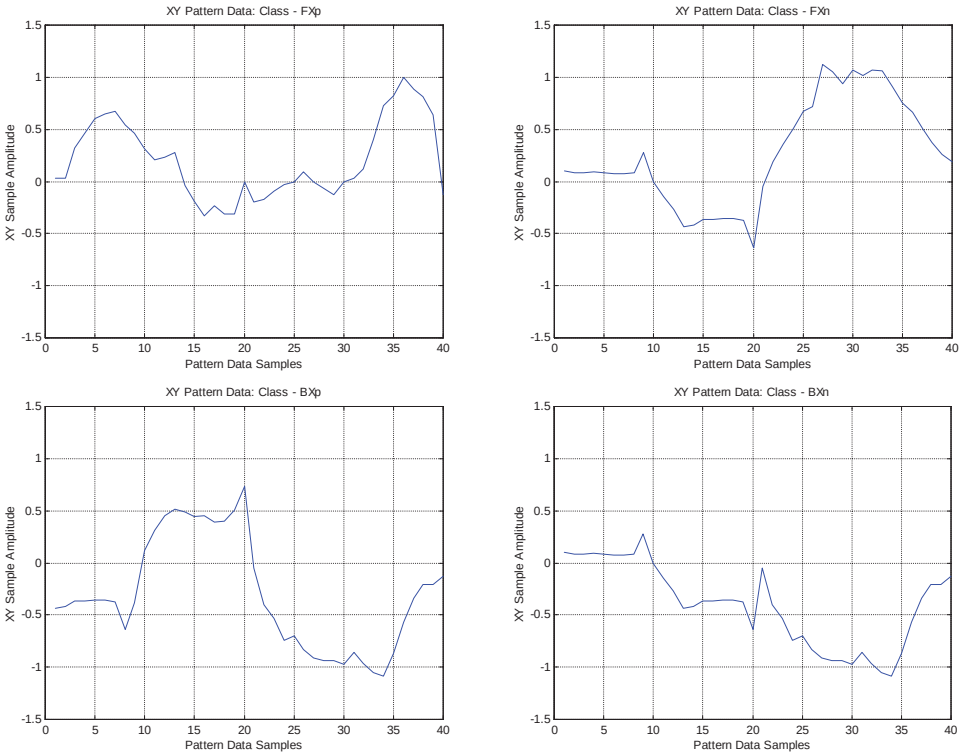

Figure 4-27: Non-Command Training Patterns – FXp, FXn, BXp, BXn

The “FXp” pattern consists of the X-axis component signal (data points 1 to 20) having amplitude greater than 0.5 and the Y-axis component signal (data points 21 to 40) having a forward command.

The “FXn” pattern consists of the X-axis component signal (data points 1 to 20) having amplitude greater less than -0.5 and the Y-axis component signal (data points 21 to 40) having a forward command.

The “BXp” pattern consists of the X-axis component signal (data points 1 to 20) having amplitude greater than 0.5 and the Y-axis component signal (data points 21 to 40) having a backward command.

The “BXn” pattern consists of the X-axis component signal (data points 1 to 20) having amplitude less than -0.5 and the Y-axis component signal (data points 21 to 40) having a backward command.

The following figure shows “LYp, LYn, RYp, and RYn” non-command signals (no examples of these signals were found in the collected training data).

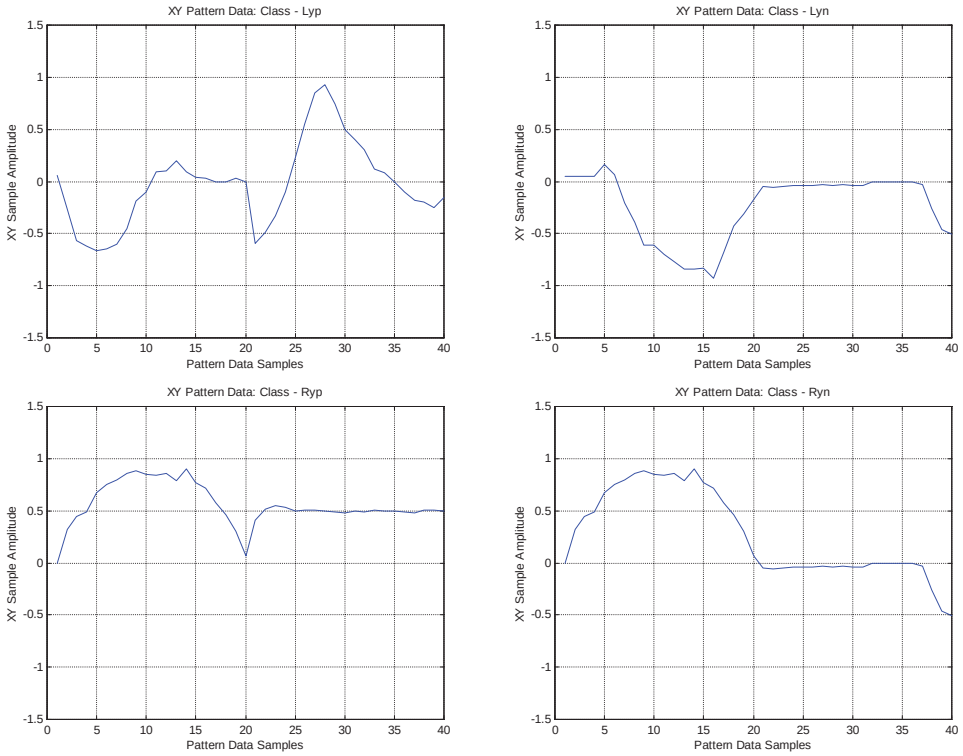

Figure 4-28: Non-Command Training Patterns – LYp, LYn, RYp, RYn

The “LYp” pattern consists of the X-axis component signal (data points 1 to 20) having a left command and the Y-axis component signal (data points 21 to 40) having amplitude greater than 0.5.

The “LYn” pattern consists of the X-axis component signal (data points 1 to 20) having a left command and the Y-axis component signal (data points 21 to 40) having amplitude less than -0.5.

The “RYp pattern consists of the X-axis component signal (data points 1 to 20) having a right command and the Y-axis component signal (data points 21 to 40) having amplitude greater than 0.5.

The “RYn” pattern consists of the X-axis component signal (data points 1 to 20) having a right command and the Y-axis component signal (data points 21 to 40) having amplitude less than -0.5.

4.6 Bootstrap Training & Test Set Creation

Once all pattern data was available the bootstrap training and test sets were created. A total of fifty bootstrap sets were created for each type of valid command (forward, backward, left and right) and non-command.

The choice of fifty bootstrap sets was based on [65] and some initial experiments where little difference was detected between 50, 100 and 1000 bootstrap sets.

Bootstrap sampling involved the following:

1. For each pattern class made up of 90 patterns randomly select with replacement a set of 90 patterns. This set becomes the bootstrap training set.
2. The patterns not selected in the training set become the test patterns. This set becomes the bootstrap test set.
3. Repeat this 50 times to create 50 bootstrap training sets and 50 bootstrap test sets.

The average number of unique patterns used for one bootstrap set is shown below.

Table 4-5: Bootstrap Training & Test Set Sizes

Class	Average Number of Unique Patterns Used in 1 Bootstrap Set		
	Training Set Patterns	Test Set Patterns	Total Patterns
Class 1 (Forward)	56	34	90
Class 2 (backward)	56	34	90
Class 3 (Left)	56	34	90
Class 4 (Right)	57	34	90
Non-Command	226	134	360

4.7 Classifier training

For each of the 50 bootstrap training sets a corresponding Neural Network (NN), Hyper-Rectangle Basis Function (HRBF), and Region and Boundary Classifier (RBC) were created and trained.

4.7.1 Neural Network Training

The neural network was trained using error back propagation (see [8] or similar texts for explanation of neural network training methods).

The hidden nodes, learning constant and mean squared error were finally set after adjustment through trial and error when training. The parameters generally settled upon were 8 hidden nodes, a learning constant of 1 and a target error of 0.01.

4.7.2 Hyper-Rectangle Basis Function Training

The major goal in training the Hyper-Rectangle basis function was to obtain the optimal amount of clusters (Hyper-Rectangles) and corresponding centres for a given percentage expansion of each Hyper-Rectangle using the lowest calculated training error. Overall, training the Hyper-Rectangle basis function consisted of the following:

1. Creation of ***n*** clusters (Hyper-Rectangles) for each command class using k-means clustering (see [60], [67], and [68]) and calculation of the centres of each Hyper-Rectangle.
2. Calculation of the minimum and maximum values for each input dimension for each Hyper-Rectangle.
3. Expanding the Hyper-Rectangles (maintaining aspect ratio) from a 0% expansion to 500% expansion.
4. Calculation of training error (ratio of total incorrect classifications to total training sets) at each expansion percentage and cluster value. The major emphasis with the calculation of error is to determine if an input is within a given Hyper-Rectangle boundary. If overlap occurs (the input is associated with two or more classes) the absolute average and maximum values are calculated for each and the class, which has the greater average and maximum values, is selected.

5. Select the corresponding Hyper-Rectangle parameters (centres, minimum and maximum values) based on the lowest training set error.

Note that the number of clusters (Hyper-Rectangles) was the same for each class as was the percentage expansion.

4.7.3 Region and Boundary Classifier Training

The major goal in training the Region and Boundary Classifier is to obtain the optimal amount of clusters (Hyper-Rectangles) and corresponding centres for a given percentage expansion of each Hyper-Rectangle using the lowest calculated training error. Overall, training the Region and Boundary Classifier is the same as the Hyper-Rectangle basis function with the addition of two "regions" which capture two types of patterns that are contained within the class Hyper-Rectangles. The training consisted of the following:

1. Creation of ***n*** clusters (Hyper-Rectangles) for each command class using k-means clustering (see [60], [67], and [68]) and calculation of the centres of each Hyper-Rectangle.
2. Calculation of the minimum and maximum values for each input dimension for each Hyper-Rectangle.
3. From all patterns labelled as forward and backward identify the maximum and minimum values for each dimension in the x-axis (points 1-20). These become the x-axis Horizontal Null Region.
4. From all patterns labelled as left and right identify the maximum and minimum values for each dimension in the y-axis (points 21-40). These become the y-axis Horizontal Null Region.
5. Merge the x-axis and y-axis horizontal null zone regions to form one "Hyper-Rectangle" defining the "Horizontal Null Region". This becomes an enabling rule for the entire classifier.
6. From all patterns labelled as forward obtain the maximum value at points 21 and 40 this becomes the Vertical Null Region (Forward). This becomes an enabling rule for the forward class Hyper-Rectangles.

7. From all patterns labelled as backward obtain the minimum value at points 21 and 40 this becomes the Vertical Null Region (Backward). This becomes an enabling rule for the backward class Hyper-Rectangles.
8. From all patterns labelled as left obtain the maximum value at points 1 and 20 this becomes the Vertical Null Region (Left). This becomes an enabling rule for the left class Hyper-Rectangles.
9. From all patterns labelled as right obtain the minimum value at points 1 and 20 this becomes the Vertical Null Region (Forward). This becomes an enabling rule for the right class Hyper-Rectangles.
10. Expand each of the Hyper-Rectangles (maintaining aspect ratio) from a 0% expansion to 500% expansion, for each incremented expansion of the Horizontal Null Region Hyper-Rectangle (from 0% to 500%). Note that the Vertical Null Region can be kept constant.
11. Calculation of training error (ratio of total incorrect classifications to total training sets) at each expansion percentage and cluster value. The major emphasis with the calculation of error is to determine if an input is within a given Hyper-Rectangle boundary. If overlap occurs (the input is associated with two or more classes) the absolute average and maximum values are calculated for each and the class, which has the greater average and maximum values, is selected.
12. Note that if a pattern is contained within the Horizontal Null Region boundary or the Vertical Null Region the resulting classification is “Null”.
13. Select the corresponding Hyper-Rectangle parameters (centres, minimum and maximum values) based on the lowest training set error.

Note that the number of clusters (Hyper-Rectangles) was the same for each class as was the percentage expansion.

4.8 Classifier Testing

Once all the classifiers were trained each classifier was tested using the corresponding bootstrap test set to calculate a measure of sensitivity and specificity for a given threshold value or percentage expansion.

From these values the receiver operating characteristics curves (ROC curve) and subsequently the area under the curve was calculated. The steps to derive these values consisted of the following:

1. For a given classifier and threshold or expansion value calculate the "apparent error" ($\overline{err}$). This consists of calculating the number of incorrect classifications made by a classifier for the corresponding bootstrap training set used to train the classifier (see [65]).
2. For a given classifier and threshold value calculate the "test error" ($\hat{E}rr^{(1)}$). This consists of calculating the number of incorrect classifications made by a classifier for the corresponding bootstrap test set (see [65]).
3. For a given classifier and threshold value calculate the "no information error rate" ($\hat{\gamma}$) for the corresponding bootstrap test set. This is the error rate if the target values for each input pattern vector were randomly assigned and an error rate was then calculated. Use the following formula (see [65]):

Equation 1

$$\hat{\gamma} = \sum_{c=1}^{C} \hat{p}_c (1 - \hat{q}_c)$$

Where ***C*** is the total number of classes for classification purposes (including the "non-command class") and $\hat{p}_c$ is the proportion (of the total number of patterns) of responses equal to a particular classification class or category and $\hat{q}_c$ is the proportion (of the total number of patterns) of required responses for each classification class or category.

4. For a given classifier and threshold value calculate the relative "over fitting rate" ($\hat{R}$) for the corresponding bootstrap test set. Use the following formula (see [65]):

Equation 2

$$\hat{R} = \frac{\hat{Err}^{(1)} - \overline{err}}{\hat{\gamma} - \overline{err}}$$

This quantity will vary from 0 with no over fitting ($\hat{Err}^{(1)} = \overline{err}$) to 1 with over fitting equal to the "no information" value ($\hat{\gamma} - \overline{err}$).

5. For a given classifier and threshold value calculate the 0.632 Estimator for the corresponding bootstrap test set. Use the following formula:

Equation 3

$$\hat{Err}^{(.632)} = 0.368\,\overline{err} + 0.632\,\hat{Err}^{(1)}$$

6. For a given classifier and threshold value calculate the 0.632+ Estimator for the corresponding bootstrap test set. Use the following formula (see [65]):

Equation 4

$$\hat{Err}^{(.632+)} = \hat{Err}^{(.632)} + (\hat{Err}^{(1)'} - \overline{err})\frac{0.368 \times 0.632\hat{R}'}{1 - 0.368\,\hat{R}'}$$

If $\hat{\gamma} \le \overline{err}$ or $\overline{err} < \hat{\gamma} \le \hat{Err}^{(1)}$, where $\hat{R}$ may fall outside (1,0), we modify the definitions to:

$\hat{Err}^{(1)'} = \min(\hat{Err}^{(1)}, \hat{\gamma})$

And

$$\hat{R}' = \begin{cases} (\hat{Err}^{(1)} - \overline{err}) / (\hat{\gamma} - \overline{err}) & \text{if } \hat{Err}^{(1)}, \hat{\gamma} > \overline{err} \\ 0 & \text{otherwise} \end{cases}$$

7. For a given classifier and threshold value calculate the sensitivity using the 0.632+ Estimator calculated from the corresponding bootstrap test set for class1 (forward), class2 (backward), class3 (left) and class4 (right). Use the following formula:

Equation 5

$$sensitivity = 1 - \frac{\left(class1\ \hat{Err}^{(.632+)} + class2\ \hat{Err}^{(.632+)} + class3\ \hat{Err}^{(.632+)} + class4\ \hat{Err}^{(.632+)}\right)}{4}$$

8. For a given classifier and threshold value calculate the specificity using the 0.632+ Estimator calculated from the corresponding bootstrap test set for non-commands (labelled class5 in the equation below).

Equation 6

$$specificity = 1 - class5\ \hat{Err}^{(.632+)}$$

9. For each classifier and a given threshold value calculate the sensitivity and specificity for each bootstrap test set and obtain a mean value for both measures across all bootstrap sets.

10. For each classifier type and a given threshold value determine the percentile 95% two sided CI by identifying the 2^{nd} and 49^{th} percentile in the sensitivity and specificity calculations across all bootstrap test sets. Note that the percentile method for confidence interval calculation is a conservative estimator (see [50] for detail of confidence intervals for the 0.632+ Estimator) and is preferred here. Also note that the Bias Corrected and Accelerated method for confidence interval estimation mentioned in [50] was tried here and found to be mostly equivalent to the percentile method.

11. For each classifier type construct ROC curves using the mean sensitivity and specificity for each threshold value and calculate the area under the curve (using the trapezoidal rule, see [69] or any mathematics text on how to calculate area under a curve). To construct the ROC curve confidence limits, use the lower and upper 95% CI sensitivity and specificity values at each threshold.

12. For each classifier type and at each threshold value calculate the overall accuracy using the following formula:

Equation 7

$$accuracy = \frac{(sensitivity + specificity)}{2}$$

13. For each classifier type identify the threshold value that provides the best overall accuracy and identify the optimal mean sensitivity, specificity and the corresponding 95% two-sided confidence intervals at that threshold.

4.8.1 Accuracy Estimation for Individual Classes

Looking at each class separately and using the ROC analysis to determine an optimal threshold value, a measure of accuracy for each class was calculated. This measure consisted of a separate 0.632+ bootstrap calculation for sensitivity averaged with the overall specificity measure for the specific classifier.

4.9 Classifier comparisons

Once all measures are calculated for each classifier, comparisons can be made using the 95% two-sided confidence intervals for the ROC area, optimal sensitivity, optimal specificity, and ranges for each of the stated measures.

4.9.1 Classifier Comparison Scoring & Ranking System

When making comparisons a scoring and ranking system is used with a tally of points for a particular classifier being made with a corresponding rank. Four outcomes are used in the comparison:

1. Equivalent: which means there is no statistical difference between the two and no points are awarded (“0”).
2. One is better: which means there is a statistical difference between the two and one point is awarded (“1”).
3. One is worse: which means there is a statistical difference between the two and a negative point is awarded (“-1”).
4. NA: This means “Not Applicable” as there is no comparison being made.

Note, that comparisons between classifiers are made using the 95% two-sided confidence interval.

4.10 Discussion and Conclusion

The application of the 0.632+ Bootstrap to ROC analysis produces an effective method for the creation of sensitivity and specificity measures, which take into account the effects of finite sampling, small sample sizes, creation of both test and training sets.

From these measures ROC curves can be generated and the area under the curve calculated, which are acknowledged as effective measures for comparing algorithms. Additionally, a method to "rank" each algorithm based on the result of comparisons with other algorithms enables differentiation between each.

The combination of all three methods provides a robust way to create training and test sets with subsequent comparison and ranking.

Also, in carrying out the Bootstrap and ROC analysis attention is drawn to the effort in creating "effective boundaries" through the generation of "representative data". As part of the pattern set creation (see 4.5) seventeen different types of non-command data were identified as "outside the boundary" and subsequently included in the training and test set data.

It is this "representative data" that is the foundation for creating "effective boundaries", which is at the heart of ANN performance.

Chapter 5 - Implicit and Explicit Boundaries for ANN

5.1 Methodology

To address the problem as stated (see Section 3.2.5) an interpretation of the problem and subsequent conceptual solution was discussed that focused on the idea of "effective boundaries". These effective boundaries are a result of implicit or explicit boundary formation influenced by comprehensive representative training (see Section 3.3.2).

An effective boundary is one that gives accuracy and predictability to the classifier, although an implicit boundary by definition is a boundary that falls between two known areas and is not explicitly known. Therefore, an implicit boundary does not truly address predictability; only the explicit boundary can fully address predictability.

With regard to an ANN, as boundary formation is arbitrary an effective boundary will always be an implicit boundary, unless every single point that the ANN will see is provided in the training set. Therefore, the use of an ANN in an application will always have an associated uncertainty due to the inherent nature of an implicit boundary.

It is this uncertainty, especially in applications that can be considered critical, such as wheelchair control using head movement [2-4] that is being addressed in this thesis.

To create effective boundaries and hence improve generalisation performance it was proposed to use representative training data, which consisted of complementary data either side of an implied boundary as the training data (see Section 3.3.1). This complementary training data takes the form of both command and non-command data, or positive and negative training examples.

In the work by [7], it was proposed that this negative training data is generated on the boundary of a hyper-cube and added to the training set, to help boundary formation and improve generalisation performance. Although, this example is simple, it is important as it proposes that the complementary training data improves generalisation performance.

Also, of interest is that the proposed solution, randomly selecting points on a hyper-cube boundary, could be considered "artificial" rather than "real" as these points may

never be seen by the ANN and therefore the formed boundaries may not be truly effective.

If we examine the idea of selecting data points on a hyper-cube, which are outside the boundary of the original training data, and then add to the original training set, the question is asked what is it really doing? Fundamentally, it is an attempt to encapsulate the training data with complementary data points to enable the ANN to form a defined boundary. The use of a geometric form such as a "box" as the reference is more about ease of use as it gives a reference position for selection of such data points rather than being a "superior" boundary form.

In work by [7], this was considered as a way to deal with the arbitrary placement of boundaries by the ANN. However, some questions were raised regarding how many points should there be and how far away from the training data should the geometric boundary be? Both of these are valid questions were not answered, as the examples used were simple, two-dimensional, and most importantly artificial.

However, if one accepts that data points selected on a geometric boundary could give improved performance by assisting in the creation of effective boundaries, why couldn't this idea be taken one step further? Why couldn't the geometric boundary be used as the basis for classification? The answer to this question although rhetorical is of course it could be. Once an explicit boundary is defined and accepted for use, this is now a defined boundary.

ANN and Data Points on a Geometric Boundary

The generation of complementary training data on a hyper-geometric boundary requires that the boundary be generated, but in doing so the boundary itself could be used as a classifier. To explain this, if we were able to generate every possible point on our hyper-geometric boundary, then the class data would be fully encapsulated, with the resulting ANN being equivalent to the hyper-geometric boundary acting as a classifier.

As can be seen in the figure below, the ANN could easily be replaced with a hyper-geometric classifier and return the same result.

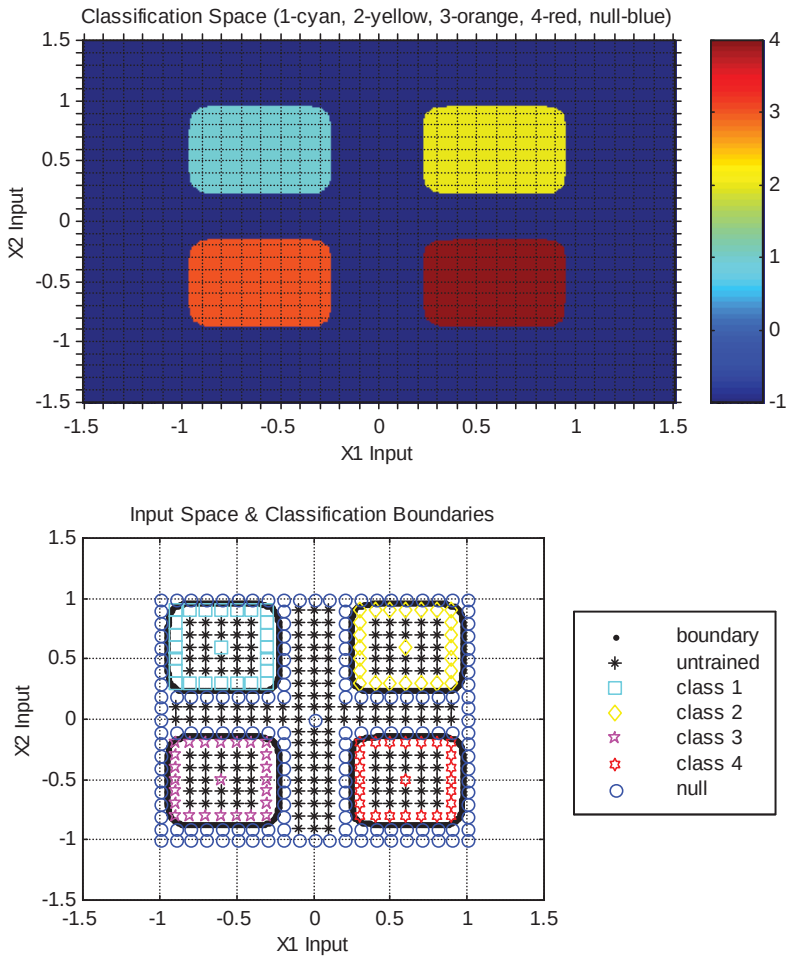

Figure 5-1: Encapsulation - NN Equivalence with Hyper-Geometric Classifier

However, this approach is artificially constructed to demonstrate the concept and raises more questions than it answers due to the artificial nature of the example.

One of the questions raised in this artificial example is how far away from the boundary of the training data should the geometric boundary be expanded? In a real example the training data only represents partial data that makes up the likely inputs for the ideal class set and realistically the boundary will need to be expanded until it reaches known points that are not considered part of that class (see 3.3.1.3.2).

In doing so, the resulting boundary will capture those likely inputs that are part of the class set and exclude those inputs that are not. This type of "optimisation" should provide a boundary with good performance. However, as discussed previously,

accuracy performance is dependent on the representativeness of the training data to enable effective boundaries to be formed.

Given these concepts, can it be applied to a more complex application, such as wheelchair control using head movement [2-4]?

Summary

In summarising, the intention of this chapter is to show that a training set containing "data outside" the boundary of the original training set provides for a more "representative" training set and enables an ANN to form effective boundaries implicitly.

The derivation of this "outside the boundary data" is from real and artificial sources. The real data is extracted from collected data, which has identified 17 "non-command" data types (see 4.4.2), and the artificial data is from data extracted from the boundary of an optimised Hyper-Rectangle.

Also, as an extension of work by others the optimised Hyper-Rectangle is used as the classification function to test the overriding assumption that the defined geometric boundary used to collect extra training data is a reasonable boundary definition.

5.1.1 The Questions Posed

Within the context of our critical application, head movement classification (used in wheelchair control), the questions posed are:

1. *Does the addition of data derived from collected data, but outside the boundary of a normal training set provide more representative data by improving the performance of an Artificial Neural Network classifier?*
2. *Does the addition of data derived on the boundary of an optimised Hyper-Rectangle, but outside the boundary of a normal training set provide more representative data by improving the performance of an Artificial Neural Network classifier?*
3. *Of the two representative data sets augmented with data outside the boundary, one derived from real data and the other from artificial data, which provides better performance when used to train an Artificial Neural Network Classifier?*

4. *Does the optimised Hyper-Rectangle when used as a classification function provide improved performance over an Artificial Neural Network Classifier?*

5.1.2 The Experiment

To answer these questions we compare the performance of three Artificial Neural Network (ANN) classifiers and one optimised Hyper-Rectangle Basis Function classifier (HRBF):

- One ANN trained using conventionally assembled training data consisting of command class data only;
- One ANN trained using both command and non-command data, with the non-command obtained from collected samples;
- One ANN trained using both command and non-command data, with the non-command data obtained from a random selection on the boundary of a Hyper-Rectangle; and
- One HRBF trained using both command and non-command data, with the non-command data obtained from collected samples.

5.1.3 Terminology Explanations

A conventionally trained ANN typically uses "positive" samples for training patterns rather than "negative" samples. This makes sense as we only care about the positives i.e. those patterns that belong to a particular class, rather than those that do not and this is the conventional approach. However, in training an ANN the resulting boundaries are arbitrarily placed not precisely fixed around the training data and if our classifier is required to discriminate between "positive" input patterns and "negative" input patterns then we have the potential to incorrectly associate or classify a given pattern.

For this experiment, training patterns that are "commands" (forward, backward, left, and right head movements) are considered as our "positive" samples and all other training patterns are considered as "non-commands" or "negative" samples.

The terms “command” and “non-command” will be used to describe the “positive” and “negative” input data, with the term “outside the boundary” referring to “non-command” or “negative” data.

5.2 Experimental Method

For more detail regarding the method used for this experiment, see Chapter 4 - Bootstrap and ROC Analysis for Head Movement.

5.2.1 Overview

As an overview this experiment employed the following process and methodology:

- Obtain training data from real subjects (see 4.4.2);
- Create training sets consisting of forward, backward, left, and right command patterns (see 4.5);
- Create a training set consisting of those patterns considered as non-command patterns (see 4.5);
- Create bootstrap training and test sets (see 4.6);
- Create and train one neural network using training data created from command classes only and train the other using both command and non-command training data (see 4.7.1);
- Create and train one optimised Hyper-Rectangle basis function using training data created from command and non-command training data (see 4.7.2);
- Using the Hyper-Rectangle basis function optimised boundaries randomly select points on the boundary to create a non-command patterns training set for an Artificial Neural Network;
- Create and train one neural network using training data created with both command and the Hyper-Rectangle non-command training data (see 4.7.1);
- Test all neural networks and optimised Hyper-Rectangle using the 0.632+ Estimator and ROC Area Under the Curve method (see 4.8); and

- Compare all neural networks and the Hyper-Rectangle to determine if a statistically significant difference in performance is obtained (see 4.9).

5.2.2 Classifier Identification

The definition of each classifier type and the acronym used to describe the classifier is shown below. The acronyms are used throughout the thesis and tables for brevity.

Table 5-1: Classifier Acronym Definitions

ACRONYM	CLASSIFIER TYPE	TRAINING DATA USED	
		Command Data	Non-Command Data
NN1	Artificial Neural Network – trained with additional real non-command data	Yes	Yes
NN2	Artificial Neural Network – trained with additional Hyper-Rectangle boundary non-command data	Yes	Yes
NN3	Artificial Neural Network – trained with only command data	Yes	No
HRBF1	Hyper-Rectangle Basis Function – trained with additional real non-command data	Yes	Yes

5.3 Results

Here we present the classification results for each of the algorithm types trained with and without the addition of the non-command training data. For each algorithm we present the calculated 0.632+ Estimator for the sensitivity and specificity and from these values the receiver operating characteristics area under the curve value is calculated. Additionally, each classifier type is compared to see if any one particular classifier algorithm shows superior or improved performance over the other.

5.3.1 ROC Analysis

5.3.1.1 Results Data

The following sections contain the tabulated test results data for each classifier, with a final comparison and discussion.

5.3.1.1.1 NN1

The test results data for the Artificial Neural Network (NN1) trained with additional data "outside the boundary" of the normal training set is shown below. This additional data was obtained from an additional seventeen types of non-command data identified in the collected data.

Table 5-2: NN1 ROC Results

TYPE	ROC (area)			Sensitivity			Specificity		
	LCI (95%)	MEAN	UCI (95%)	LCI (95%)	MEAN	UCI (95%)	LCI (95%)	MEAN	UCI (95%)
NN1	0.9196	0.9660	0.9948	0.8928	0.9333	0.9672	0.9034	0.94	0.9795

Table 5-3: NN1 ROC Accuracy Results

TYPE	Accuracy		
	LCI (95%)	MEAN	UCI (95%)
NN1	0.8981	0.9367	0.9734

Table 5-4: NN1 ROC Range Results

TYPE	95% Range (UCI - LCI)		
	ROC	Sensitivity	Specificity
NN1	0.0752	0.0744	0.0761

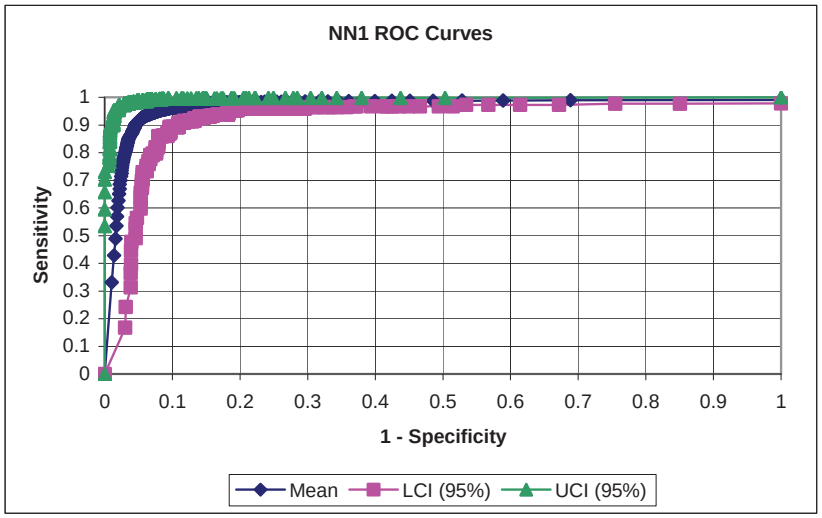

Figure 5-2: NN1 ROC Curves

5.3.1.1.2 NN2

The test results data for the Artificial Neural Network (NN2) trained with additional data "outside the boundary" of the normal training set is shown below. This additional data was obtained from the boundary of an optimised Hyper-Rectangle basis function.

Table 5-5: NN2 ROC Results

TYPE	ROC (area)			Sensitivity			Specificity		
	LCI (95%)	MEAN	UCI (95%)	LCI (95%)	MEAN	UCI (95%)	LCI (95%)	MEAN	UCI (95%)
NN2	0.6731	0.8294	0.9503	0.8389	0.9132	0.9635	0.6053	0.7072	0.8189

Table 5-6: NN2 ROC Accuracy Results

TYPE	Accuracy		
	LCI (95%)	MEAN	UCI (95%)
NN2	0.7221	0.8102	0.8912

Table 5-7: NN2 ROC Range Results

TYPE	95% Range (UCI - LCI)		
	ROC	Sensitivity	Specificity
NN2	0.2772	0.1246	0.2136

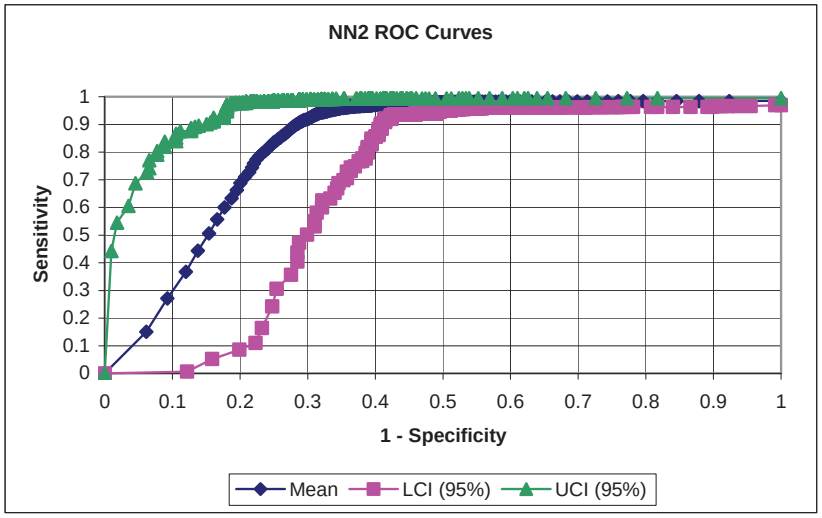

Figure 5-3: NN2 ROC Curves

5.3.1.1.3 NN3

The test results data for the Artificial Neural Network (NN3) trained with only the normal training set is shown below. This training set consists of only the class data (forward, backward, left, and right).

Table 5-8: NN3 ROC Results

TYPE	ROC (area)			Sensitivity			Specificity		
	LCI (95%)	MEAN	UCI (95%)	LCI (95%)	MEAN	UCI (95%)	LCI (95%)	MEAN	UCI (95%)
NN3	0.6468	0.7999	0.9314	0.8042	0.8714	0.9262	0.6087	0.6710	0.7355

Table 5-9: NN3 ROC Accuracy Results

TYPE	Accuracy		
	LCI (95%)	MEAN	UCI (95%)
NN3	0.7065	0.7712	0.8309

Table 5-10: NN3 ROC Range Results

TYPE	95% Range (UCI - LCI)		
	ROC	Sensitivity	Specificity
NN3	0.2846	0.1220	0.1268

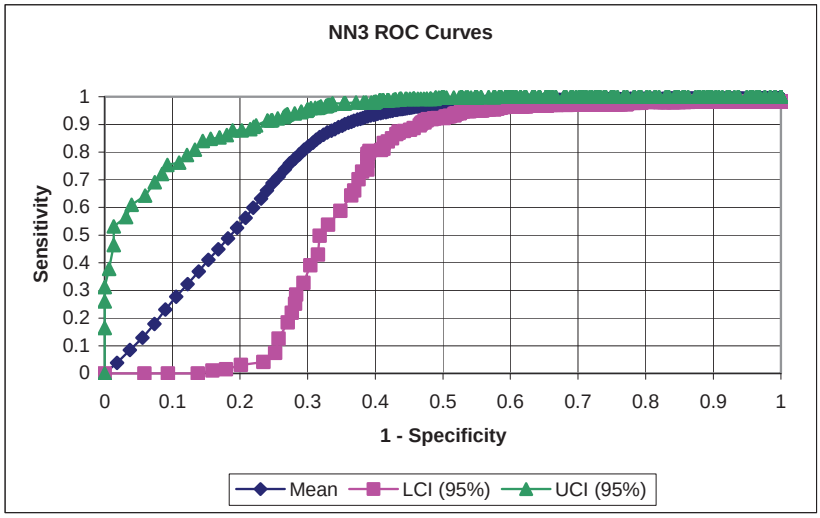

Figure 5-4: NN3 ROC Curves

5.3.1.1.4 HRBF1

The test results data for the optimised Hyper-Rectangle Basis Function (HRBF1) trained with additional data "outside the boundary" of the normal training set is shown below.

Table 5-11: HRBF1 ROC Results

TYPE	ROC (area)			Sensitivity			Specificity		
	LCI (95%)	MEAN	UCI (95%)	LCI (95%)	MEAN	UCI (95%)	LCI (95%)	MEAN	UCI (95%)
HRBF1	0.8590	0.9030	0.9414	0.8987	0.9533	0.9833	0.8293	0.852	0.874

Table 5-12: HRBF1 ROC Accuracy Results

TYPE	Accuracy		
	LCI (95%)	MEAN	UCI (95%)
HRBF1	0.8640	0.9027	0.9287

Table 5-13: HRBF1 ROC Range Results

TYPE	95% Range (UCI - LCI)		
	ROC	Sensitivity	Specificity
HRBF1	0.0823	0.0846	0.0447

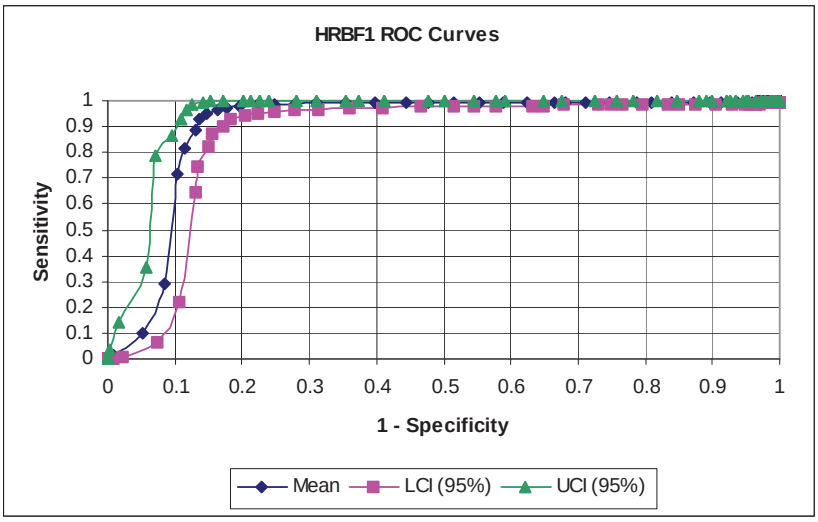

Figure 5-5: HRBF1 ROC Curves

5.3.1.1.5 All ROC Results

The tables shown below contain the test results for each of the classifiers.

Table 5-14: All ROC Results

TYPE	ROC (area)			Sensitivity			Specificity		
	LCI (95%)	MEAN	UCI (95%)	LCI (95%)	MEAN	UCI (95%)	LCI (95%)	MEAN	UCI (95%)
NN1	0.9196	0.9660	0.9948	0.8928	0.9333	0.9672	0.9034	0.94	0.9795
HRBF1	0.8590	0.9030	0.9414	0.8987	0.9533	0.9833	0.8293	0.852	0.874
NN2	0.6731	0.8294	0.9503	0.8389	0.9132	0.9635	0.6053	0.7072	0.8189
NN3	0.6468	0.7999	0.9314	0.8042	0.8714	0.9262	0.6087	0.6710	0.7355

Table 5-15: All ROC Accuracy Results

TYPE	Accuracy		
	LCI (95%)	MEAN	UCI (95%)
NN1	0.8981	0.9367	0.9734
HRBF1	0.8640	0.9027	0.9287
NN2	0.7221	0.8102	0.8912
NN3	0.7065	0.7712	0.8309

Table 5-16: All ROC Range Results

TYPE	95% Range (UCI - LCI)		
	ROC	Sensitivity	Specificity
NN1	0.0752	0.0744	0.0761
HRBF1	0.0823	0.0846	0.0447
NN2	0.2772	0.1246	0.2136
NN3	0.2846	0.1220	0.1268

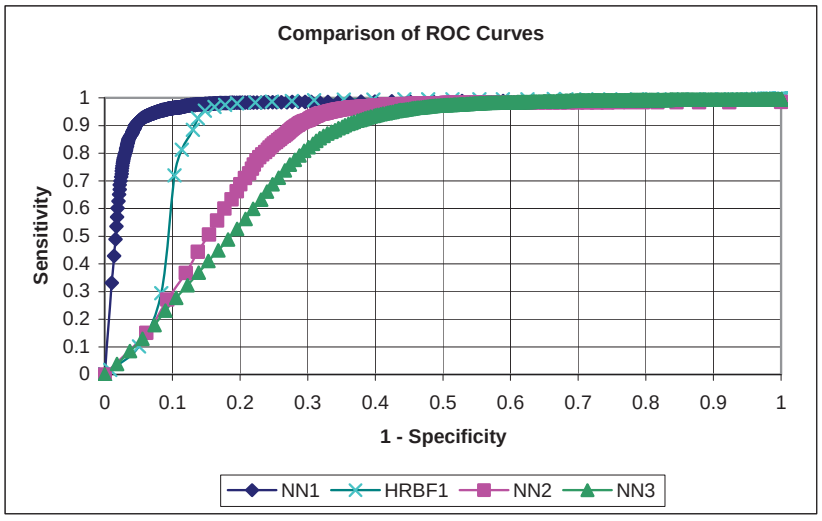

Figure 5-6: Comparison of ROC Curves

5.3.1.2 ROC Comparison & Ranking

The comparison matrix below shows that the results of the comparison of ROC test results for each of the classifiers.

Table 5-17: All ROC Results

TYPE	ROC (area)			Sensitivity			Specificity		
	LCI (95%)	MEAN	UCI (95%)	LCI (95%)	MEAN	UCI (95%)	LCI (95%)	MEAN	UCI (95%)
NN1	0.9196	0.9660	0.9948	0.8928	0.9333	0.9672	0.9034	0.94	0.9795
HRBF1	0.8590	0.9030	0.9414	0.8987	0.9533	0.9833	0.8293	0.852	0.874
NN2	0.6731	0.8294	0.9503	0.8389	0.9132	0.9635	0.6053	0.7072	0.8189
NN3	0.6468	0.7999	0.9314	0.8042	0.8714	0.9262	0.6087	0.6710	0.7355

Table 5-18: ROC Comparison Matrix

ROC (area)						
TYPE	NN1	HRBF1	NN2	NN3	Points	Rank
NN1	NA	Equivalent	Equivalent	Equivalent	0	1
HRBF1	Equivalent	NA	Equivalent	Equivalent	0	1
NN2	Equivalent	Equivalent	NA	Equivalent	0	1
NN3	Equivalent	Equivalent	Equivalent	NA	0	1
Sensitivity						
TYPE	NN1	HRBF1	NN2	NN3	Points	Rank
NN1	NA	Equivalent	Equivalent	Equivalent	0	1
HRBF1	Equivalent	NA	Equivalent	Equivalent	0	1
NN2	Equivalent	Equivalent	NA	Equivalent	0	1
NN3	Equivalent	Equivalent	Equivalent	NA	0	1
Specificity						
TYPE	NN1	HRBF1	NN2	NN3	Points	Rank
NN1	NA	NN1 is better	NN1 is better	NN1 is better	3	1
HRBF1	HRBF1 is worse	NA	HRBF1 is better	HRBF1 is better	1	2
NN2	NN2 is worse	NN2 is worse	NA	Equivalent	-2	3
NN3	NN3 is worse	NN3 is worse	Equivalent	NA	-2	3

Table 5-19: ROC Ranking Matrix

TYPE	ROC	Sensitivity	Specificity	Total Points	Rank
NN1	0	0	3	3	1
HRBF1	0	0	1	1	2
NN2	0	0	-2	-2	3
NN3	0	0	-2	-2	3

5.4 Discussion and Conclusion

One of the goals of this chapter was to assess the findings of [7] in which training data "outside the boundary" of a conventionally assembled training set is used to improve the performance of an ANN classifier. The question of whether this idea used in a more complex and critical context, such as wheelchair control, would improve the performance of a neural network classifier was posed because the previous work by others used simple two-dimensional problems for the algorithm assessment. Further to this, artificially derived data rather than data derived from a real-time signal was used as the basis of this "outside the boundary" data.

Using data collected from real subjects and creating patterns for both commands and non-commands, three different ANN and one HRBF were created and tested using the 0.632+ Estimator and ROC area under the curve analysis.

Results from the comparison show that the ANN trained with additional data "outside the boundary" (command and non-command data) out performs an ANN trained with a conventional training set (command data only). Examination of this result highlights a significant improvement in the mean specificity of the neural network trained with this additional data (~27% improvement).

For the ANN trained with additional data "outside the boundary" (command and non-command data) obtained on the boundary of the HRBF there was no statistically significant improvement in performance over an ANN trained with a conventional training set. However, examination of the data shows mean values to be slightly higher for ROC area under the curve, sensitivity, and specificity, but no statistically significant conclusions can be drawn.

For the HRBF trained with additional data "outside the boundary" (command and non-command data) there are significant improvements in all mean performances when compared with an ANN trained with a conventional training set (command data only). Examination of this result highlights a significant improvement in the mean specificity when compared to the conventionally trained ANN (~18% improvement).

Comparisons between the ANN and HRBF trained with the same data (command and non-command) mainly show statistical equivalence, with exception of specificity, and in the overall comparison the ANN (NN1) is ranked first with the HRBF ranked second.

From the results obtained in this chapter the originally posed questions can now be answered as follows:

1. *Does the addition of data derived from collected data, but outside the boundary of a normal training set provide more representative data by improving the performance of an Artificial Neural Network classifier?*

Results showed that the addition of data outside the boundary of a normal training set did improve the performance of an ANN classifier. Of interest is that there was a statistically significant improvement in mean specificity of ~30%, ~9% for mean sensitivity, and ~15% for mean ROC area under the curve.

For this to occur, fundamentally, the inclusion of this data made the training set more representative and hence there was a significant improvement in performance.

2. *Does the addition of data derived on the boundary of an optimised Hyper-Rectangle, but outside the boundary of a normal training set provide more representative data by improving the performance of an Artificial Neural Network classifier?*

Results showed that the addition of data outside the boundary of the normal training set, but derived on the boundary of an optimised Hyper-Rectangle didn't significantly improve the performance of an ANN classifier.

This shows the random selection of a limited number of data points on this boundary to be insufficient to provide representative data. It must be noted, that as the number of data points selected increases, the performance of this ANN would approach that of the optimised Hyper-Rectangle.

3. *Of the two representative data sets augmented with data outside the boundary, one derived from real data and the other from artificial data, which provides better performance when used to train an Artificial Neural Network Classifier?*

As the results show, the most representative data set was that set consisting of the real data. The ANN trained with this showed statistically significant improvements in performance. The ANN trained with the added artificially derived data did not show any significant improvements in performance.

4. *Does the optimised Hyper-Rectangle when used as a classification function provide improved performance over an Artificial Neural Network Classifier?*

The optimised Hyper-Rectangle when used as a classifier showed improved performance across all mean measured parameters with a statistically significant improvement in mean specificity of ~18%.

The Hyper-Rectangle and ANN when trained with the same data showed that the ANN was able to perform slightly better and ranked first with the HRBF ranked second overall. This shows that the Hyper-Rectangle as a classification function in this application is valid.

Conclusions

In drawing conclusions from these results and having answered the originally posed questions some important findings can be highlighted:

- That a representative training set provides an ANN with improved performance over one that is less representative;
- That a representative training set must consist of all likely inputs to the classifier, not just those patterns that form the classes to be classified;
- That a Hyper-Rectangle basis function trained with representative training data is comparable to an ANN trained with the same data (in this application); and
- Effective boundaries can be formed using implicit or explicit means as long as the training data is representative of the likely inputs to the classifier.

These findings provide backing to the proposed conceptual solutions that required effective boundaries to be formed. In this case effective boundaries were formed using both implicit means (ANN) and explicit means (HRBF).

The fact that there were effective boundaries created with the optimised Hyper-Rectangle basis function means that a solution that addresses both accuracy and predictability is available, as long as the training data is representative.

Chapter 6 - An Advanced Boundary Concept

6.1 Methodology

Results from Chapter 5 - Implicit and Explicit Boundaries for ANN, show that effective boundaries can be formed both implicitly and explicitly as long as the training data is truly representative of the likely inputs to the classifier.

In this case the likely inputs to the classifier were not just part of the command data set (forward, backward, left, and right commands), but included patterns that are considered as “non-commands” (those commands that are not forward, backward, left, or right).

These non-commands are considered as “outside the boundary” of the training set that would be used to train the classifier in a normal or conventional sense, but when put to the test the inclusion of this “outside the boundary” data to a training set brings statistically significant improvements in performance.

Of interest, was the performance of the Hyper-Rectangle, both as a source for the additional “outside the boundary” training data, and as a geometric classifier. This Hyper-Rectangle was expanded until the error was optimised and used as both a boundary from which to select training data and as a classifier itself.

In this case the addition of randomly selected data on the boundary of the Hyper-Rectangle didn’t show any statistical improvements in performance, when compared with an ANN that was trained with command data only. This result, although contrary to expectations from [7], is a confirmation of the simplicity of the example given, which when tested with a more complex problem falls over.

Also, the concept of random selection on this boundary was not a practical solution, as those selections would be unlikely to be part of the set of likely inputs to the classifier.

However, the failure here isn’t one of work in [7], it is a problem of the size of the input space that renders the concept not adequate in this case. If the number of points is increased to a point where every finite point on the Hyper-Rectangle boundary is used, then the concept is valid. Unfortunately, in this case the input space is too large for this to be realised, however the result of having every point on the Hyper-Rectangle would

be equivalent to using the geometric form as the classifier, thereby making the use of the ANN redundant.

In the case of the Hyper-Rectangle used as the classifier and trained with both command and non-command data, this approach showed improvements in mean performance and statistically significant improvements in specificity when compared with an ANN trained with command data only. Also, this Hyper-Rectangle classifier compared favourably to an ANN trained with both command and non-command data.

Importantly, results from Chapter 5 - Implicit and Explicit Boundaries for ANN, confirm the hypothesis that a representative training set must contain examples of all likely inputs to the classifier, not just the classes to be classified (in this case forward, backward, left, and right).

Also, that effective boundaries can be formed implicitly with an ANN and explicitly with a Hyper-Rectangle Basis Function optimised with the representative data.

6.1.1.1 Improving on the Optimised Hyper-Rectangle Basis Function

The Hyper-Rectangle Basis Function as shown in Chapter 5 - Implicit and Explicit Boundaries for ANN, is a valid type of geometric classifier when optimised with the representative data.

As shown below in the idealised examples the boundary of the Hyper-Rectangle, without expansion will most probably not include those other data points considered as part of the ideal class set and of the likely set of inputs

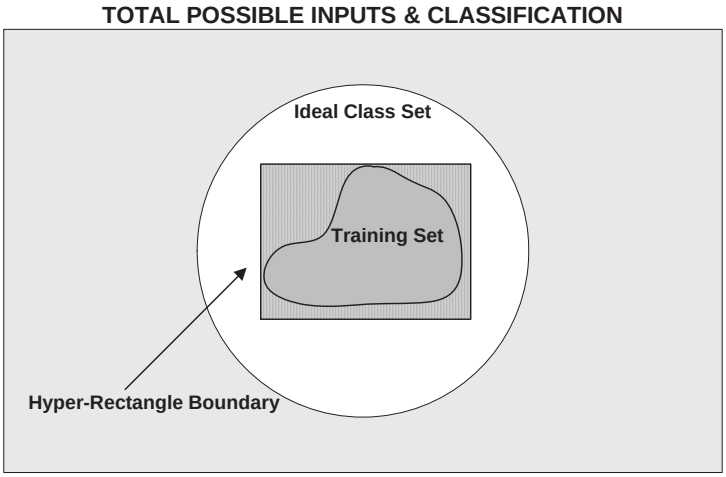

Figure 6-1: Hyper-Rectangle Boundary

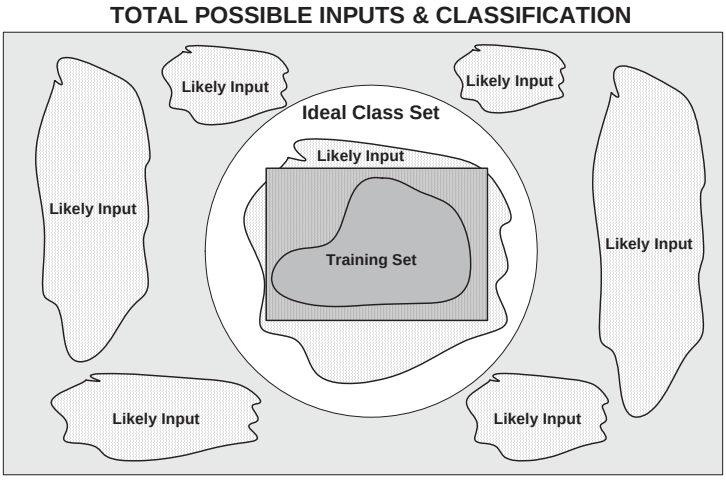

Figure 6-2: Hyper-Rectangle Boundary – with Likely Inputs

In the example above the geometric classifier could be said to be "specific", but not all that "sensitive", as it is constrained by the immediate training data used to construct the geometric boundary. However if the boundary is expanded this changes, as sensitivity increases and specificity decreases. At some point in the expansion an optimal value for sensitivity and specificity is reached.

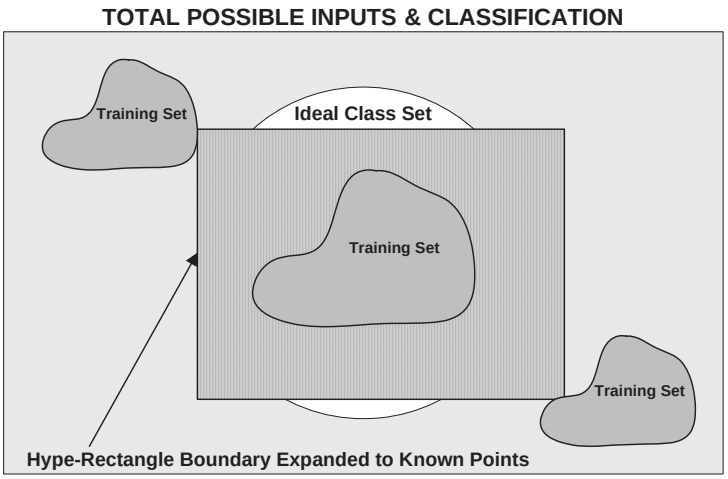

Figure 6-3: Hyper-Rectangle Boundary Expanded (Optimised) to Known Points

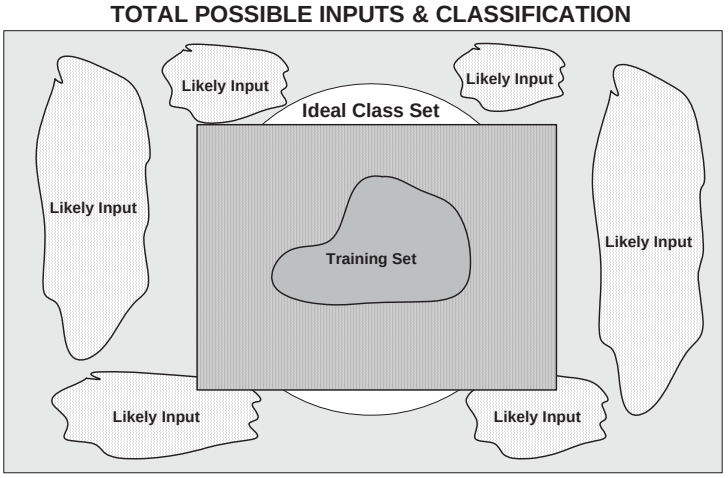

Figure 6-4: Optimised Hyper-Rectangle Boundary – with Likely Inputs

While the examples above (and results from Chapter 5) show that the Hyper-Rectangle is capable of forming effective boundaries when optimised with truly representative data and using measures of sensitivity and specificity, can it be improved?

If the conceptual figures are examined again, it appears that the expansion although optimised may include data that is not part of the class set, but part of the likely inputs. In this case improvements could be obtained by adding additional regions that would exclude such optimal expansions.

The figure below shows such an improvement, where the Hyper-Rectangle is expanded to known points (as part of an optimisation), but some regions are such as to warrant their own regions. These additional regions become exclusion zones as arbitration as to class membership is not allowed if an overlap with another class occurs.

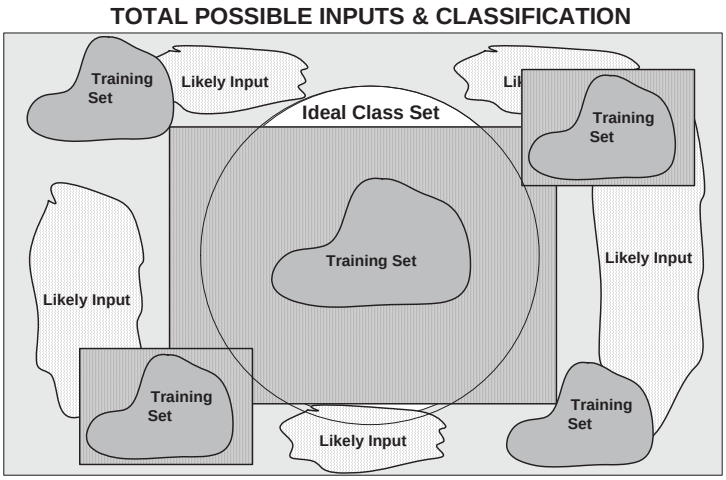

Figure 6-5: Hyper-Rectangle with Additional Regions

In examining the optimised Hyper-Rectangle Basis Function in two-dimensions, it can be seen that the geometric expansion has "overtaken" or "included" other types of signal that are considered as likely inputs to the classifier. Of these two types of signal, one is the "Null" or "Flat Line" (see Figure 4-24: Non-Command Training Pattern – Flat Line or Null), and the other is a head movement command that doesn't start or finish within the two second window, but is still captured by the class Hyper-Rectangle, due to the optimised expansion.

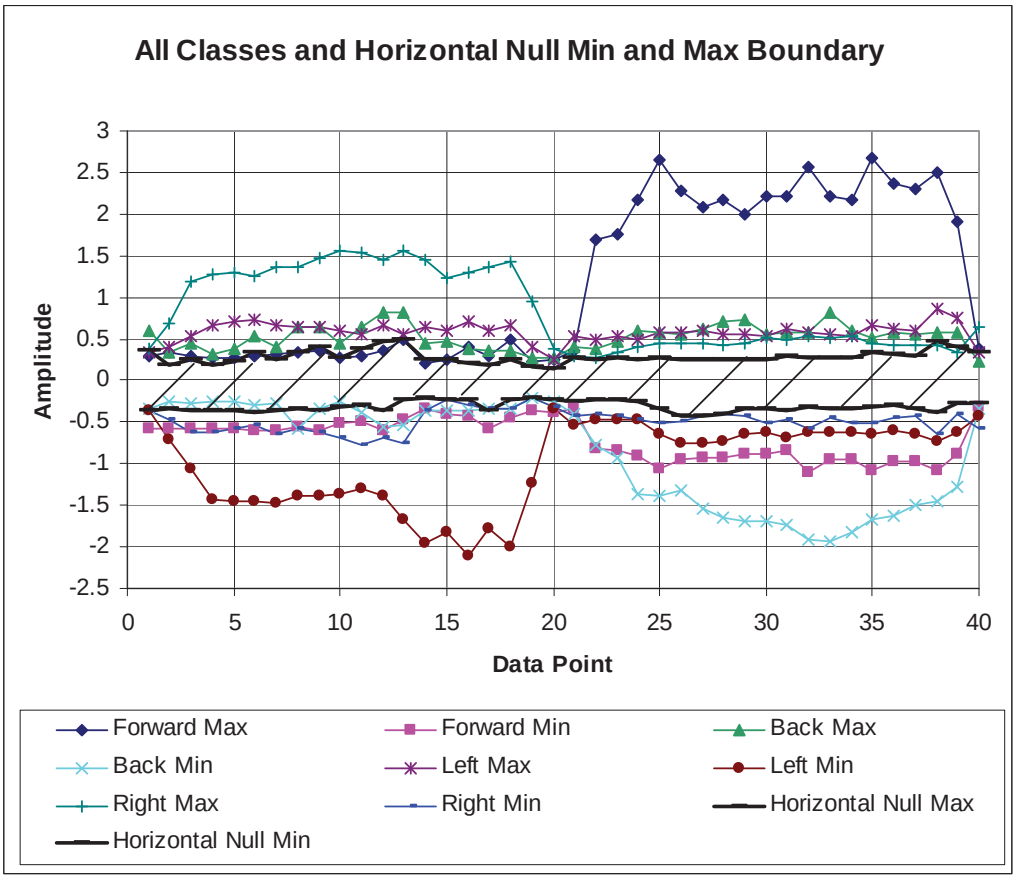

Figure 6-6: Optimised Hyper-Rectangle Basis Function – All Classes (Real Data)

If these regions are added to the optimised Hyper-Rectangle Basis Function, the classifier is no longer just an Optimised Hyper-Rectangle Basis Function, but is in fact an Optimised Region and Boundary Classifier. This description is now more correct, in that regions and boundaries are used to classify the input data, rather than just a geometric form.

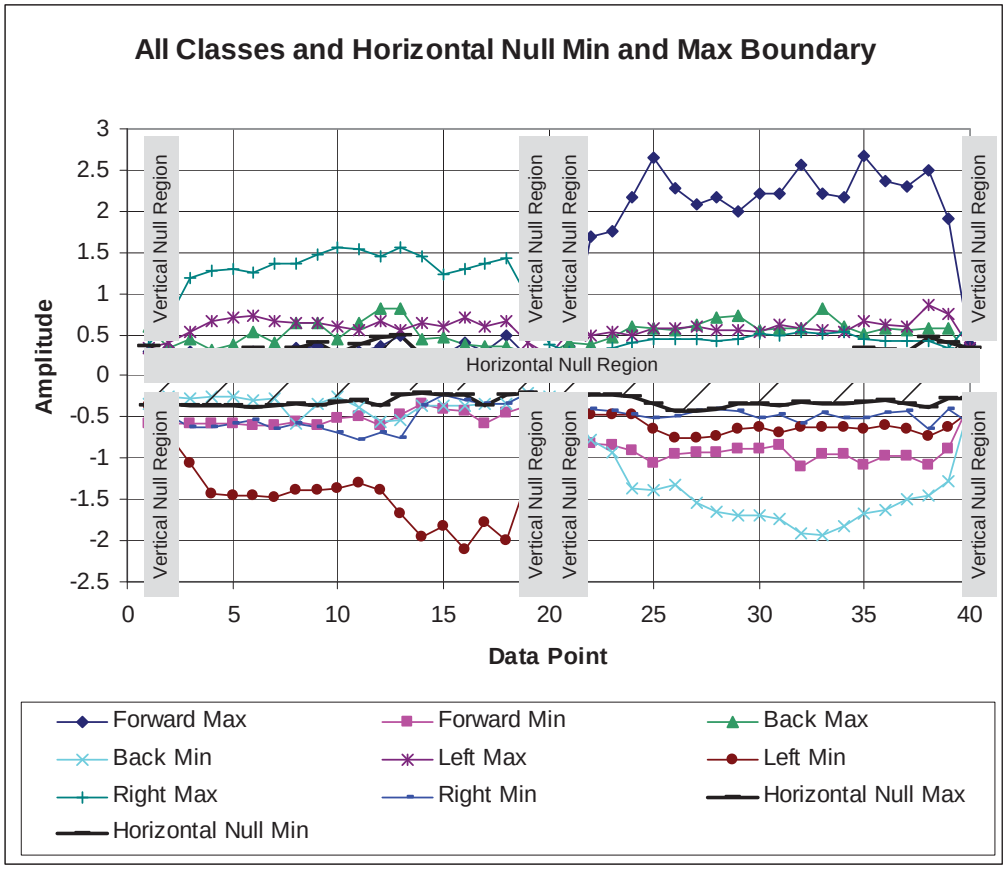

Figure 6-7: Optimised Hyper-Rectangle Basis Function with Null Regions

The architecture of such a classifier is shown below.

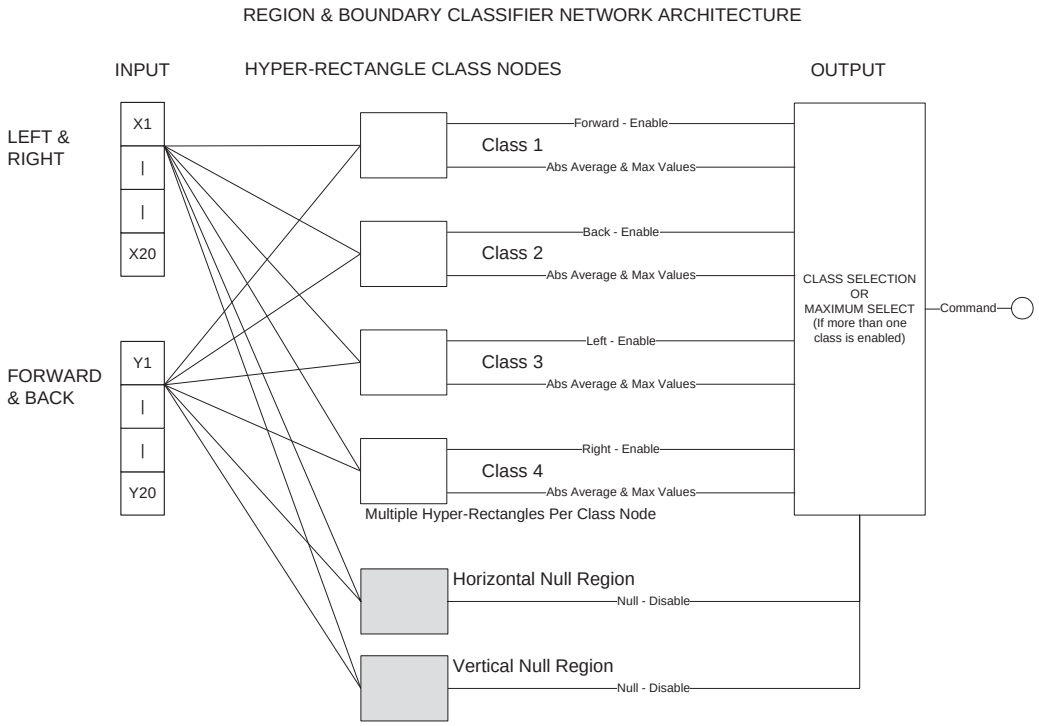

Figure 6-8: Region and Boundary Classifier Network Architecture

Summary

In summarising, the intention of this chapter is to show that an Optimised Hyper-Rectangle Basis Function further improved by the inclusion of additional null regions is in fact a better performing classifier than that of the Optimised Hyper-Rectangle Basis Function.

Also, that this Optimised Region and Boundary Classifier is as at least equivalent in performance to the ANN trained with both command and non-command data.

6.1.2 The Questions Posed

Within the context of our critical application, head movement classification (used in wheelchair control), the questions posed are:

1. *Of the two classifiers employing explicit boundaries, the Optimised Region and Boundary Classifier and the Optimised Hyper-Rectangle Basis Function, which provides better performance?*
2. *Is the Optimised Region and Boundary Classifier equivalent in performance to an Artificial Neural Network trained with command and non-command data?*
3. *Is a classifier based on an explicitly defined boundary a better performer than a classifier based on an implicitly defined boundary?*
4. *Of the classifiers tested in Chapter 5 how does the Optimal Region and Boundary Classifier compare?*

6.1.3 The Experiment

To answer this question we compare the performance of three Artificial Neural Network (ANN) classifiers and one optimised Hyper-Rectangle Basis Function classifier (HRBF), and the Optimised Region and Boundary Classifier (RBC):

- One ANN trained using conventionally assembled training data consisting of command class data only;
- One ANN trained using both command and non-command data, with the non-command obtained from collected samples;

- One ANN trained using both command and non-command data, with the non-command data obtained from a random selection on the boundary of a Hyper-Rectangle;

- One HRBF trained using both command and non-command data, with the non-command data obtained from collected samples; and

- One RBC trained using both command and non-command data, with the non-command data obtained from collected samples.

6.1.4 Terminology Explanations

A conventionally trained ANN typically uses "positive" samples for training patterns rather than "negative" samples. This makes sense as we only care about the positives i.e. those patterns that belong to a particular class, rather than those that do not and this is the conventional approach. However, in training an ANN the resulting boundaries are arbitrarily placed not precisely fixed around the training data and if our classifier is required to discriminate between "positive" input patterns and "negative" input patterns then we have the potential to incorrectly associate or classify a given pattern.

For this experiment, training patterns that are "commands" (forward, backward, left, and right head movements) are considered as our "positive" samples and all other training patterns are considered as "non-commands" or "negative" samples.

The terms "command" and "non-command" will be used to describe the "positive" and "negative" input data, with the term "outside the boundary" referring to "non-command" or "negative" data.

6.2 Experimental Method

For more detail regarding the method used for this experiment, see Chapter 4 - Bootstrap and ROC Analysis for Head Movement.

6.2.1 Overview

As an overview this experiment employed the following process and methodology:

- Obtain training data from real subjects (see 4.4.2);

- Create training sets consisting of forward, backward, left, and right command patterns (see 4.5);
- Create a training set consisting of those patterns considered as non-command patterns (see 4.5);
- Create bootstrap training and test sets (see 4.6);
- Create and train one neural network using training data created from command classes only and train the other using both command and non-command training data (see 4.7.1);
- Create and train one optimised Hyper-Rectangle basis function using training data created from command and non-command training data (see 4.7.2);
- Using the Hyper-Rectangle basis function optimised boundaries randomly select points on the boundary to create a non-command patterns training set for an Artificial Neural Network;
- Create and train one neural network using training data created with both command and the Hyper-Rectangle non-command training data (see 4.7.1);
- Create and train one Optimised Region and Boundary Classifier using training data created from command and non-command training data (see 4.7.3);
- Test all neural networks and optimised Hyper-Rectangle using the 0.632+ Estimator and ROC Area Under the Curve method (see 4.8); and
- Compare all neural networks, the Hyper-Rectangle, and the Region and Boundary classifiers to determine if a statistically significant difference in performance is obtained (see 4.9).

6.2.2 Classifier Identification

The definition of each classifier type and the acronym used to describe the classifier is shown below. The acronyms are used throughout the thesis and tables for brevity.

Table 6-1: Classifier Acronym Definitions

ACRONYM	CLASSIFIER TYPE	TRAINING DATA USED	
		Command Data	Non-Command Data
RBF1	Region and Boundary Classifier – trained with additional real non-command data	Yes	Yes
NN1	Artificial Neural Network – trained with additional real non-command data	Yes	Yes
HRBF1	Hyper-Rectangle Basis Function – trained with additional real non-command data	Yes	Yes
NN2	Artificial Neural Network – trained with additional Hyper-Rectangle boundary non-command data	Yes	Yes
NN3	Artificial Neural Network – trained with only command data	Yes	No

6.3 Results

Here we present the classification results for each of the algorithm types trained with and without the addition of the non-command training data. For each algorithm we present the calculated 0.632+ Estimator for the sensitivity and specificity and from these values the receiver operating characteristics area under the curve value is calculated. Additionally, each classifier type is compared to see if any one particular classifier algorithm shows superior or improved performance over the other.

6.3.1 ROC Analysis

6.3.1.1 Results Data

The following sections contain the tabulated test results data for each classifier, with a final comparison and discussion.

Note, that the results for NN1, NN2, NN3, and HRBF1 were obtained from Chapter 5 - Implicit and Explicit Boundaries for ANN.

6.3.1.1.1 RBC1

The test results data for the Region and Boundary Classifier (RBC1) trained with additional data "outside the boundary" of the normal training set is shown below. This additional data was obtained from an additional seventeen types of non-command data identified in the collected data.

Table 6-2: RBC1 ROC Results

TYPE	ROC (area)			Sensitivity			Specificity		
	LCI (95%)	MEAN	UCI (95%)	LCI (95%)	MEAN	UCI (95%)	LCI (95%)	MEAN	UCI (95%)
RBC1	0.9506	0.9808	0.9966	0.9302	0.9662	0.9907	0.9509	0.9733	0.9838

Table 6-3: RBC1 ROC Accuracy Results

TYPE	Accuracy		
	LCI (95%)	MEAN	UCI (95%)
RBC1	0.9406	0.9698	0.9873

Table 6-4: RBC1 ROC Range Results

TYPE	95% Range (UCI - LCI)		
	ROC	Sensitivity	Specificity
RBC1	0.0459	0.0605	0.0329

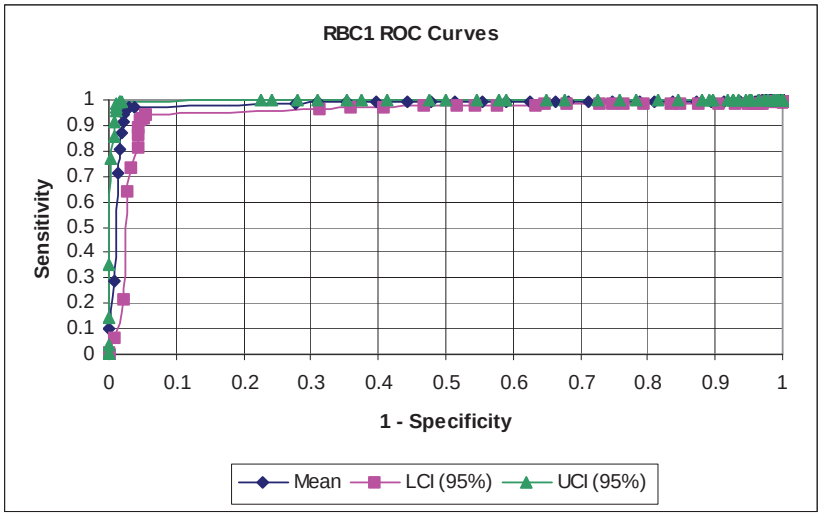

Figure 6-9: RBC1 ROC Curves

6.3.1.1.2 All ROC Results

The tables shown below contain the test results for each of the classifiers.

Table 6-5: All ROC Results

TYPE	ROC (area)			Sensitivity			Specificity		
	LCI (95%)	MEAN	UCI (95%)	LCI (95%)	MEAN	UCI (95%)	LCI (95%)	MEAN	UCI (95%)
RBC1	0.9506	0.9808	0.9966	0.9302	0.9662	0.9907	0.9509	0.9733	0.9838
NN1	0.9196	0.9660	0.9948	0.8928	0.9333	0.9672	0.9034	0.94	0.9795
HRBF1	0.8590	0.9030	0.9414	0.8987	0.9533	0.9833	0.8293	0.852	0.874
NN2	0.6731	0.8294	0.9503	0.8389	0.9132	0.9635	0.6053	0.7072	0.8189
NN3	0.6468	0.7999	0.9314	0.8042	0.8714	0.9262	0.6087	0.6710	0.7355

Table 6-6: All ROC Accuracy Results

TYPE	Accuracy		
	LCI (95%)	MEAN	UCI (95%)
RBC1	0.9406	0.9698	0.9873
NN1	0.8981	0.9367	0.9734
HRBF1	0.8640	0.9027	0.9287
NN2	0.7221	0.8102	0.8912
NN3	0.7065	0.7712	0.8309

Table 6-7: All ROC Range Results

TYPE	95% Range (UCI - LCI)		
	ROC	Sensitivity	Specificity
RBC1	0.0459	0.0605	0.0329
NN1	0.0752	0.0744	0.0761
HRBF1	0.0823	0.0846	0.0447
NN2	0.2772	0.1246	0.2136
NN3	0.2846	0.1220	0.1268

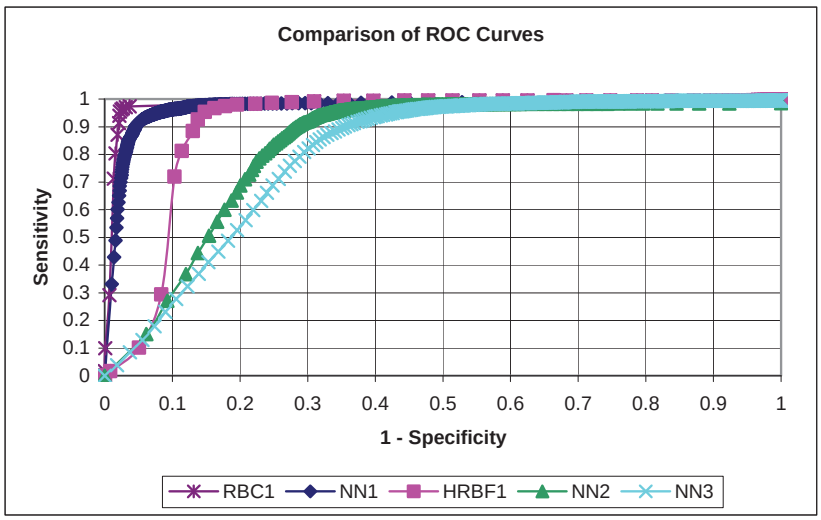

Figure 6-10: Comparison of ROC Curves

6.3.1.2 ROC Comparison & Ranking

The comparison matrix below shows that the results of the comparison of ROC test results for each of the classifiers.

Table 6-8: All ROC Results

TYPE	ROC (area)			Sensitivity			Specificity		
	LCI (95%)	MEAN	UCI (95%)	LCI (95%)	MEAN	UCI (95%)	LCI (95%)	MEAN	UCI (95%)
RBC1	0.9506	0.9808	0.9966	0.9302	0.9662	0.9907	0.9509	0.9733	0.9838
NN1	0.9196	0.9660	0.9948	0.8928	0.9333	0.9672	0.9034	0.94	0.9795
HRBF1	0.8590	0.9030	0.9414	0.8987	0.9533	0.9833	0.8293	0.852	0.874
NN2	0.6731	0.8294	0.9503	0.8389	0.9132	0.9635	0.6053	0.7072	0.8189
NN3	0.6468	0.7999	0.9314	0.8042	0.8714	0.9262	0.6087	0.6710	0.7355

Table 6-9: ROC Comparison Matrix

ROC (area)							
TYPE	RBC1	NN1	HRBF1	NN2	NN3	Points	Rank
RBC1	NA	Equivalent	RBC1 is better	RBC1 is better	RBC1 is better	3	1
NN1	Equivalent	NA	Equivalent	Equivalent	Equivalent	0	2
HRBF1	HRBF1 is worse	Equivalent	NA	Equivalent	Equivalent	-1	3
NN2	NN2 is worse	Equivalent	Equivalent	NA	Equivalent	-1	3
NN3	NN3 is worse	Equivalent	Equivalent	Equivalent	NA	-1	3
Sensitivity							
TYPE	RBC1	NN1	HRBF1	NN2	NN3	Points	Rank
RBC1	NA	Equivalent	Equivalent	Equivalent	RBC1 is better	1	1
NN1	Equivalent	NA	Equivalent	Equivalent	Equivalent	0	2
HRBF1	Equivalent	Equivalent	NA	Equivalent	Equivalent	0	2
NN2	Equivalent	Equivalent	Equivalent	NA	Equivalent	0	2
NN3	NN3 is worse	Equivalent	Equivalent	Equivalent	NA	-1	3
Specificity							
TYPE	RBC1	NN1	HRBF1	NN2	NN3	Points	Rank
RBC1	NA	Equivalent	RBC1 is better	RBC1 is better	RBC1 is better	3	1
NN1	Equivalent	NA	NN1 is better	NN1 is better	NN1 is better	3	1
HRBF1	HRBF1 is worse	HRBF1 is worse	NA	HRBF1 is better	HRBF1 is better	0	2
NN2	NN2 is worse	NN2 is worse	NN2 is worse	NA	Equivalent	-3	3
NN3	NN3 is worse	NN3 is worse	NN3 is worse	Equivalent	NA	-3	3

Table 6-10: ROC Ranking Matrix

TYPE	ROC	Sensitivity	Specificity	Total Points	Rank
RBC1	3	1	3	7	1
NN1	0	0	3	3	2
HRBF1	-1	0	0	-1	3
NN2	-1	0	-3	-4	4
NN3	-1	-1	-3	-5	5

6.4 Discussion and Conclusion

One of the goals of this chapter was to assess how the Optimised Region and Boundary Classifier performs when compared with the other types of classifiers assessed in Chapter 5 - Implicit and Explicit Boundaries for ANN.

Using data collected from real subjects and creating patterns for both commands and non-commands, three different ANN, one HRBF, and one RBC were created and tested using the 0.632+ Estimator and ROC area under the curve analysis.

Results from the comparison show that the RBC has a statistically significant improvement in ROC (area) performance (~8%), when compared with the HRBF, which backs the hypothesis that the additional regions used by the Region and Boundary Classifier indeed improve performance.

When the RBC is compared with the ANN trained with additional data "outside the boundary" (command and non-command data) there is no statistically significant improvements in performance and can be considered equivalent. However, there are slight improvements of between ~2% to ~3% across all mean measured parameters and examination of the ROC curve (Figure 6-10: Comparison of ROC Curves) also shows the RBC to be a better performing algorithm.

This result shows that boundaries formed either implicitly or explicitly with the same representative data can be considered equivalent from a statistical point of view. However, the explicit boundary in this case the RBC, shows improved mean performances across all major measures and when each algorithm is compared and ranked accordingly, the RBC is seen as the best performer.

From the results obtained in this chapter the originally posed questions can now be answered as follows:

1. *Of the two classifiers employing explicit boundaries, the Optimised Region and Boundary Classifier and the Optimised Hyper-Rectangle Basis Function, which provides better performance?*

The RBC shows statistically significant improvements in ROC (area) mean performance when compared with the HRBF (~8%).

Of interest here is that the RBC provides statistically significant improvements in specificity with a mean improvement of ~12%. Note, that this improvement in specificity is due to the effect of the additional "Horizontal and Vertical Null Regions", as would be expected.

2. *Is the Optimised Region and Boundary Classifier equivalent in performance to an Artificial Neural Network trained with command and non-command data?*

The direct comparison of the RBC and ANN (NN1), show statistical equivalence. However, the RBC has higher mean performance in all measures (ROC area, sensitivity, and specificity) and could be considered a better performer. This assertion is backed up by resulting ranking, in which the RBC is ranked 1st and the ANN (NN1) ranked 2nd.

3. *Is a classifier based on an explicitly defined boundary a better performer than a classifier based on an implicitly defined boundary?*

The RBC and ANN (NN1) were trained on the same data and are considered statistically equivalent. Therefore, in this case it can be said that the explicit and implicit boundary formed from the same representative data are equivalent.

However, the RBC when compared with all the other algorithms tested here ranks 1st and could be considered the best performer. In this case the explicit boundary out performs the implicit boundary.

4. *Of the classifiers tested in Chapter 5 how does the Optimal Region and Boundary Classifier compare?*

Of all the classifiers tested the RBC is ranked 1st, the ANN trained with both command and non-command data ranked 2nd, the HRBF ranked 3rd, the ANN trained with both command and non-command data obtained from the boundary of a HRBF ranked 4th, and the ANN trained with command data only ranked 5th.

Clearly the RBC provides improved performance when compared with the other classifiers, with both good sensitivity and specificity.

Conclusions

In drawing conclusions from these results and having answered the originally posed questions some important findings can be highlighted:

- That the RBC is the best performing and ranked classifier when compared with the ANN classifiers and the HRBF;
- That the RBC as an example of an explicitly formed boundary is at least as effective as an implicitly formed boundary, and when ranking is considered implies that in this case the explicitly formed boundary is a better performer; and
- That the RBC as an improvement on the HRBF is validated as a better performing classifier, with statistically significant improvements in performance.

These findings provide the backing to the final solution to the problem posed, as the Optimised Region and Boundary Classifier provide both an effective boundary and a boundary that is known.

When looking at the original problem, the solution must address accuracy and predictability, which the Optimised Region and Boundary Classifier is able to do. Noting, the training data must be representative of all the likely inputs to the classifier, which in this case it is.

Chapter 7 - Conclusion

7.1 Summary

The motivation for this thesis began with the safety of algorithms used in critical applications in complex environments (see Section 1.1), especially for head movement classification (as used to control a powered wheelchair). In this particular application an ANN is used as the classification algorithm to decode head movements into commands (forward, backward, left, and right) and non-commands.

However, the use of an ANN is not without uncertainties, especially when presented with data "outside the boundary" of the training set. It was within this context that the problem statement (see Section 1.2) for this thesis was made and was stated as:

"How to ensure the accuracy and predictability of a classifier, used for head movement classification, for data points outside the boundary of the training set."

The key words from this problem statement are "accuracy", "predictability", and "outside the boundary" and were reached through a process using "five whys" where questions are asked until a root cause is reached (see 3.2).

In conducting the "five whys" a problem tree was established, which identified the problems and potential causes (see 3.2) and with further analysis was trimmed (see 3.2.3) until a final problem tree was established (see 3.2.4). This final problem tree identified the problems and the affected characteristics of a classifier that was then transformed into the problem statement.

Once the problem was established in this top-down logical manner, where the problems are associated with the defining characteristics of accuracy and predictability a bottom-up conceptual solution could be developed that also affected these defining characteristics (see 3.3)

Conceptually, the solution lay with the training data being "representative" and the boundaries being "explicitly" or "implicitly" defined (see 3.4). However, what is really required is that the decision boundaries are effective. For the decision boundaries to be effective the training data must be such as to allow boundaries to be formed in a manner

that is effective (see 3.3.1). Whether a boundary is defined implicitly or explicitly, effectiveness relies on the training data.

A training set that is representative all types of data that are "likely inputs" must be used and the data must be complementary enough to enable boundaries to be formed (see 3.3.1). The likely inputs must also include that data that isn't included in the classes to be classified i.e. "outside the boundary", but are likely inputs to the classifier.

Conceptually, the issue of whether implicit or explicit boundaries are best from an accuracy point of view is dependant on the training data, but when both are provided with the same data, they can be considered equivalent (in this case).

However, as the problem is also concerned with predictability an implicit boundary still provides uncertainty, as the boundaries are not known. So although the implicit boundary may provide accuracy (due to the training set being representative), it will never be known: therefore only the explicitly defined boundary can truly address the issue of predictability.

In realising the concept of representative training data and using head movement classification (as used in powered wheelchair control) as the critical application, sample data was collected from volunteers (see 4.4.2). From this collected data, there were four command patterns (forward, backward, left, and right) and seventeen types of non-command patterns were identified that would need to be included in the training/test sets for this data to be considered truly representative (see 4.5.2).

In realising the implicit and explicit boundaries an ANN was selected for the implicit boundaries (see 4.3.1) and a Hyper-Rectangle Basis Function was used as the basis of the explicit boundary (see 4.3.2).

The selection of the Hyper-Rectangle was an extension of the work by [6, 7], in which a hyper-box was used to define a boundary around training data that becomes a source for additional data. This additional training data is added to the original set, thereby providing a type of encapsulation (see Figure 5-1) that would limit the boundary of the original class data. Unfortunately, the example given was based on a simple two-dimensional artificial dataset and when put to the test in the application used here is not a valid method (see 5.3.1.2).

However, by making the initial assumption that the now encapsulated training data is a valid boundary also implies that the boundary if realised with a geometric function is also a valid classification boundary. Therefore, the use of the hyper-box would be valid as the classifier rather than the ANN. This assumption was validated as true through testing in this case (see 5.3.1.2).

In realising a geometric boundary around the training data, effective boundaries need to be formed and this formation would require that some form of generalisation be implemented, such as the expansion of the boundary to an optimal point. A key word here is "optimal" as the expansion must be controlled and an optimal point reached.

The method chosen for the optimisation is based on the Receiver Operating Characteristics (ROC) which uses sensitivity and specificity measures obtained from a sliding threshold (expansion %) to characterise the algorithm (see 4.2). This allows for an optimal expansion point to be chosen that considers both sensitivity and specificity.

In using the Hyper-Rectangle as a basis for classification that is then optimised with ROC methodology, further optimisation is possible with the use of an additional "Horizontal and Vertical Null Zone" (see 4.3.3). These additional zones provide coverage for those patterns that the Hyper-Rectangle includes within its boundary, but are not part of that class set.

By adding these additional zones or regions the classifier is actually using not just "Hyper-Rectangles" to classify, but is really just using "Regions and Boundaries". If these regions and boundaries are expanded to the optimal point using the ROC methodology, we have an Optimised Region and Boundary Classifier.

It is this Optimised Region and Boundary Classifier that fulfils the requirements of the solution necessary to answer the initial problem question. These requirements are that the classifier must be accurate and predictable. By having truly representative training data with an optimally chosen expansion point (optimal generalisation), accuracy is addressed, and by having an explicit boundary predictability is also addressed.

Hypotheses Tested

In proposing a solution to the problem (see 2.4.1), a validation of the hypothesised solutions and underlying assumptions were required. The initially proposed hypotheses with answers are shown below:

1. *That the addition of data consisting of likely inputs to the classifier, but outside the boundary of a normal class training set makes the training set more representative.*

The results of this test confirmed that the additional data obtained from "non-commands", which were also likely inputs to the classifier, made for a more representative data set (see Chapter 5 - Implicit and Explicit Boundaries for ANN).

The result of a "more representative" training set is improved performance over one that is "less representative".

2. *That the addition of data consisting of points randomly selected on the boundary of a Hyper-Rectangle, but outside the boundary of a normal class training set makes the training set more representative.*

The results of this test confirmed that the use of the optimised Hyper-Rectangle as a source for additional training data did not make the training set more representative (see Chapter 5 - Implicit and Explicit Boundaries for ANN).

3. *That the use of an explicitly defined boundary such as that obtained from an Optimised Hyper-Rectangle is a valid method for classifying data.*

The results of this test showed that that this classifier was comparable to the best ANN (see Chapter 5 - Implicit and Explicit Boundaries for ANN), but not completely statistically equivalent. However, the ANN showed only slight improvements in performance when compared with the Hyper-Rectangle and highlights the fact that the Hyper-Rectangle is a valid method.

4. *That an Optimised Hyper-Rectangle Classifier can be improved by the addition of regions and boundaries to form a better classifier known as an Optimised Region and Boundary Classifier.*

The results of this test show that the RBC overall is a better performing classifier (see 6.3.1.2).

5. *That an explicitly defined boundary performs better than an implicitly defined boundary when trained on the same data.*

The results of this test show that for the best performing explicit boundary (RBC) and the best performing implicit boundary (ANN) both are statistically equivalent, however the explicit boundary had better mean performances over all measures, and is considered the best classifier with the highest ranking (see 6.3.1.2).

6. *That the Optimised Region and Boundary Classifier is the highest ranked classifier when compared to an Artificial Neural Network or an Optimised Hyper-Rectangle Classifier.*

The results of this test show that the highest ranked classifier was the Optimised Region and Boundary Classifier (RBC) with the ANN trained on the same data ranked second (see 6.3.1.2).

A Final Solution

In conducting these tests and answering questions, the final solution to the originally stated problem could be forthcoming. To ensure the accuracy and predictability of a classifier used for head movement classification (as used to control a powered wheelchair) the classifier must be trained on truly representative data obtained from real samples including those samples “outside of the boundary” of a typical training set and the boundary of the classifier must be known.

The Optimised Region and Boundary Classifier when trained on truly representative training data provides for a more accurate and predictable classifier as shown in this case.

7.2 Significance

The significance of the work undertaken in this body of research and subsequent thesis within the context of head movement classification (for the purpose of powered wheelchair control) consists of the following:

1. *The creation of a novel algorithm that addresses both accuracy and predictability when required to classify data outside the boundary of the training set.*

In this case the Optimised Region and Boundary Classifier uses explicit boundaries to provide predictability and accuracy is achieved with boundaries which are optimally placed using training data that is representative of all inputs expected to the classifier and is complementary i.e. consists of data either side of an implied boundary.

This algorithm addresses both accuracy and predictability and can be seen as the best performer when compared to other algorithms such as an ANN and a Hyper-Rectangle.

The significance of this is that it is potentially an inherently safer classification algorithm.

2. *The creation of a novel algorithm that could be used in Medical Devices where issues of safety are important.*

As this Optimised Region and Boundary Classifier addresses issues of accuracy and predictability, it is potentially an inherently a "safer" algorithm to use. With this in mind, Medical Devices that use classifier algorithms (2-D real time amplitude varying signals) may benefit from the use of this algorithm, especially if the classification algorithm has the potential to impact patient safety or well being.

3. *A definition and demonstration of what constitutes a representative training set.*

A representative training set can be defined as a set that consists of samples of all types of likely inputs to the classifier. Also, of significance is that these samples should be complementary (exist either side of an implied boundary), which enables the formation of effective boundaries.

In the example in this thesis, beyond the four "positive" command signals (forward, back, left, and right) an additional seventeen types of "negative" non-command signals that the classifier could see were identified and used to create a representative data set.

Hence representative data is not restricted to only four "types" of head movement signal, but is extended to at least an additional seventeen "types" of signal in this case.

4. *A demonstration that the use of artificial data randomly derived on the boundary of a Hyper-Cube does not make the training data more representative.*

Randomly selected data on the boundary of a Hyper-Cube did not significantly improve the performance of an ANN. However, as the number of samples on the boundary of the Hyper-Cube increases the performance will approach that of the Hyper-Cube being the classification algorithm itself.

Intuitively, this makes sense as in simple examples all data on the boundary could be included in the training set, but in more complex applications such as the one used in this thesis it is not practical to do so. However, one way to create an equivalent classifier that includes all or most data on the boundary of the Hyper-Cube is to use the Hyper-Cube itself as the classifier.

5. *A demonstration that boundaries formed implicitly (ANN) and those formed explicitly (RBC) are equivalent when provided with the same training data.*

When the two classifiers (RBC and an ANN) were trained with the same representative data to enable effective boundary formation, a statistically equivalent performance was achieved.

6. *The creation of a novel method to test the classification algorithms termed in this thesis as "The 0.632+ Bootstrap ROC Analysis".*

This involved a novel combination of existing methods known as the "0.632+ Bootstrap" and "ROC" analysis techniques, where measures of sensitivity and specificity are obtained for a given threshold.

The merging of these two methods saw fifty bootstrap sets created and values of sensitivity and specificity for each threshold increment calculated for each bootstrap set. From these values the mean sensitivity and specificity was obtained along with percentile based confidence intervals.

Based on the calculated measures of sensitivity and specificity for each classifier type tested, comparisons were made, which used a point scoring system to enable a ranking

based on each of the measured parameters (ROC Area Under the Curve, Sensitivity, and Specificity). From this an overall ranking was possible.

7.3 Future Research

Given the problem addressed by this thesis and the application, areas for future research that could be looked at include the following:

1. *The use of the Optimised Region and Boundary Classifier in Medical Devices that employ a 2-D real time amplitude varying signal.*

Generally speaking, Medical Devices have a heavier burden to provide safe operation when compared with other types of devices due to the impact of malfunction or erroneous results on the safety and well being of patients. This being the case, the use of a classification algorithm that may impact the safety or wellbeing of a patient is of concern.

In this case the use of the Optimised Region and Boundary Classifier may provide for increased certainty regarding "accuracy" and "predictability" and may be a valid algorithm to use in suitable applications.

One such application worth mentioning (from previous professional experience) is the detection of "Pump Suck Down" in a Left Ventricle Assist Device (LVAD), which is a life giving device typically given to patients suffering end stage heart failure as a bridge to transplant or as an end stage therapy.

The term "Pump Suck Down" refers to a state whereby the tubing at the input to the pump (venous line) collapses due to the negative pressure (and other physiological interactions) and an unstable state of pump speed control ensues. This type of situation is potentially life threatening and a means to classify this state to enable a control correction is required. Note, that this state is detectable from the current (amps) waveform of the pump.

Obviously, such a classification algorithm must be "safe", which requires that the algorithm be both "accurate" and "predictable". As such the use of the Optimised Region and Boundary Classifier is a potential solution in this case.

2. *The application of "what constitutes an effective boundary" to other types of classification problems such as letter or face recognition.*

In this thesis the data examined was 2-D real time obtained from two accelerometers mounted on a users head, hence "head movement" classifier. One of the problems addressed was the issue of effective boundary placement, which relies on representative data and controlled boundary placement (through the use of complementary data).

Taking this idea into other domains such as letter or face recognition would test whether this approach is valid in other domains and also potentially provide for a unified definition of what constitutes effective boundary placement.

3. *Improving on the Optimised Region and Boundary Classifier when used to classify head movement.*

In this thesis the classifiers were set up to indicate commands (forward, back, left, or right) and non-commands (all signals that are not classified commands). Also, in creating a "representative" data set seventeen non-commands were identified, but these were not differentiated as the classifiers only indicated five possible states i.e. forward, back, left, right, or non-command.

Another possible architecture may be to enable the classifier to classify all known types of signal (for this thesis there were 21), plus enable an "indeterminate" where the signals do not exceed a threshold.

This type of set up could provide for a type of classifier that identifies "unknowns", which could later be analysed, labelled, and included in the training set. This type of classifier would to some extent be adaptive.

Note that this idea of adaptivity could also be used with an ANN.

BIBLIOGRAPHY

[1] FDA, "Guidance for Industry, FDA Reviewers and Compliance on Off-The-Shelf Software Use in Medical Devices," 1999.

[2] T. Joseph and H. Nguyen, "Neural network control of wheelchairs using telemetric head movement," presented at Engineering in Medicine and Biology Society, 1998. Proceedings of the 20th Annual International Conference of the IEEE, 1998.

[3] P. Taylor, H. Nguyen, and A. Craig, "Head Movement Recognition for Power Wheelchair Control," presented at Engineering and Physical Sciences in Medicine 2002, Rotorua, New Zealand, 2002.

[4] P. B. Taylor and H. T. Nguyen, "Performance of a Head-Movement Interface for Wheelchair Control," presented at 25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2003, Cancun, Mexico, 2003.

[5] ISO, "ISO 14971:2000(E) Medical devices — Application of risk management to medical devices," 2000.

[6] D. Chakraborty and N. R. Pal, "Making a multilayered perceptron network say - "don't know" when it should," presented at Proceedings of the 9th International Conference on Neural Information Processing, 2002.

[7] D. Chakraborty and N. R. Pal, "A novel training scheme for multilayered perceptrons to realize proper generalization and incremental learning," *Neural Networks, IEEE Transactions on*, vol. 14, pp. 1-14, 2003.

[8] J. M. Zurada, *Introduction to Artificial Neural Systems*. Boston, MA: PWS Publishing Company, 1995.

[9] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359-366, 1989.

[10] Z. Shi and V. Cherkassky, "Factors controlling generalization ability of MLP networks," presented at International Joint Conference on Neural Networks, 1999.

[11] G. P. Zhang, "Avoiding Pitfalls in Neural Network Research," *Systems, Man and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 37, pp. 3-16, 2007.

[12] J. Yulei, "Uncertainty in the output of artificial neural networks," *Medical Imaging, IEEE Transactions on*, vol. 22, pp. 913-921, 2003.

[13] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights," presented at International Joint Conference on Neural Networks, 1990.

[14] D. Sarkar, "Randomness in generalization ability: a source to improve it," *Neural Networks, IEEE Transactions on*, vol. 7, pp. 676-685, 1996.

[15] Y. Grandvalet and S. Canu, "Comments on Noise injection into inputs in back propagation learning," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 25, pp. 678-681, 1995.

[16] L. Holmstrom and P. Koistinen, "Using additive noise in back-propagation training," *Neural Networks, IEEE Transactions on*, vol. 3, pp. 24-38, 1992.

[17] J. Sietsma and R. J. F. Dow, "Creating artificial neural networks that generalize," *Neural Networks*, vol. 4, pp. 67-79, 1991.

[18] M. Skurichina, S. Raudys, and R. P. W. Duin, "k-nearest neighbors directed noise injection in multilayer perceptron training," *Neural Networks, IEEE Transactions on*, vol. 11, pp. 504-511, 2000.

[19] P. Niyogi, F. Girosi, and T. Poggio, "Incorporating prior information in machine learning by creating virtual examples," *Proceedings of the IEEE*, vol. 86, pp. 2196-2209, 1998.

[20] L. Breiman, "Bagging Predictors," *Machine Learning*, vol. 24, pp. 123-140, 1996.

[21] C. Mingmin and L. Bruzzone, "A semilabeled-sample-driven bagging technique for ill-posed classification problems," *Geoscience and Remote Sensing Letters, IEEE*, vol. 2, pp. 69-73, 2005.

[22] R. E. Schapire, "The strength of weak learnability," *Machine Learning*, vol. 5, pp. 197-227, 1990.

[23] R. E. Schapire, P. Bartlett , Y. Freund , and W. S. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," *Annals of Statistics*, vol. 26, pp. 1651, 1998.

[24] V. Boyarshinov and M. Magdon-Ismail, "Efficient Optimal Linear Boosting of a Pair of Classifiers," *Neural Networks, IEEE Transactions on*, vol. 18, pp. 317-328, 2007.

[25] Y. Freund, "Boosting a Weak Learning Algorithm by Majority," *Information and Computation*, vol. 121, pp. 256-285, 1995.

[26] I. Cohen, F. G. Cozman, N. Sebe, M. C. Cirelo, and T. S. Huang, "Semisupervised learning of classifiers: theory, algorithms, and their application to human-computer interaction," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, pp. 1553-1566, 2004.

[27] M. T. Fardanesh and O. K. Ersoy, "Classification accuracy improvement of neural network classifiers by using unlabeled data," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 36, pp. 1020-1025, 1998.

[28] B. M. Shahshahani and D. A. Landgrebe, "The effect of unlabeled samples in reducing the small sample size problem and mitigating the Hughes phenomenon," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 32, pp. 1087-1095, 1994.

[29] L. Breiman, "Randomizing Outputs to Increase Prediction Accuracy," *Machine Learning*, vol. 40, pp. 229-242, 2000.

[30] M. M. Adankon, M. Cheriet, and A. Biem, "Semisupervised Least Squares Support Vector Machine," *Neural Networks, IEEE Transactions on*, vol. 20, pp. 1858-1870, 2009.

[31] C. Hong and L. Luoqing, "Semisupervised Multicategory Classification With Imperfect Model," *Neural Networks, IEEE Transactions on*, vol. 20, pp. 1594-1603, 2009.

[32] L. K. Hansen and P. Salamon, "Neural network ensembles," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 12, pp. 993-1001, 1990.

[33] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas, "On combining classifiers," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, pp. 226-239, 1998.

[34] R. Rifkin and A. Klautau, "In defense of one-vs-all classification," *Machine Learning Research*, vol. 5, pp. 101, 2004.

[35] A. Rego da Rocha Neto and G. de Alencar Barreto, "WCI 04 On the Application of Ensembles of Classifiers to the Diagnosis of Pathologies of the Vertebral Column: A Comparative Analysis," *Latin America Transactions, IEEE (Revista IEEE America Latina)*, vol. 7, pp. 487-496, 2009.

[36] L. Breiman, "Arcing Classifiers," *The Annals of Statistics*, vol. 26, pp. 801-824, 1998.

[37] B. Gabrys and A. Bargiela, "General fuzzy min-max neural network for clustering and classification," *Neural Networks, IEEE Transactions on*, vol. 11, pp. 769-783, 2000.

[38] H. Ishibuchi and M. Nii, "Fuzzification of input vectors for improving the generalization ability of neural networks," presented at Fuzzy Systems Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on, 1998.

[39] P. K. Simpson, "Fuzzy min-max neural networks. I. Classification," *Neural Networks, IEEE Transactions on*, vol. 3, pp. 776-786, 1992.

[40] J. T. Kwok, I. W. H. Tsang, and J. M. Zurada, "A Class of Single-Class Minimax Probability Machines for Novelty Detection," *Neural Networks, IEEE Transactions on*, vol. 18, pp. 778-785, 2007.

[41] J. S. Lim, "Finding Features for Real-Time Premature Ventricular Contraction Detection Using a Fuzzy Neural Network System," *Neural Networks, IEEE Transactions on*, vol. 20, pp. 522-527, 2009.

[42] L. Chulhee and D. A. Landgrebe, "Decision boundary feature extraction for neural networks," *Neural Networks, IEEE Transactions on*, vol. 8, pp. 75-83, 1997.

[43] G. N. Karystinos and D. A. Pados, "On overfitting, generalization, and randomly expanded training sets," *Neural Networks, IEEE Transactions on*, vol. 11, pp. 1050-1057, 2000.

[44] S. G. Baker, "The Central Role of Receiver Operating Characteristic (ROC) Curves in Evaluating Tests for the Early Detection of Cancer," *J Natl Cancer Inst*, vol. 95, pp. 511-515, 2003.

[45] L. Arbach, J. M. Reinhardt, D. L. Bennett, and G. Fallouh, "Mammographic masses classification: comparison between backpropagation neural network (BNN), K nearest neighbors (KNN), and human readers," presented at Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Canadian Conference on, 2003.

[46] W. T. Hung, H. T. Nguyen, W. B. Lee, M. T. Rickard, B. S. Thornton, and A. Blinowska, "Diagnostic abilities of three CAD methods for assessing

microcalcifications in mammograms and an aspect of equivocal cases decisions by radiologists," *Australasian Physical & Engineering Sciences in Medicine*, vol. 26 number 3, pp. 78 - 83, 2003.

[47] D. Faraggi, B. Reiser, and E. F. Schisterman, "ROC curve analysis for biomarkers based on pooled assessments," *Statistics in Medicine. 2003 Aug 15;22(15):2515-27*, 2003.

[48] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, pp. 1145-1159, 1997.

[49] W. A. Yousef, R. F. Wagner, and M. H. Loew, "Comparison of non-parametric methods for assessing classifier performance in terms of ROC parameters," presented at Applied Imagery Pattern Recognition Workshop, 2004. Proceedings. 33rd, 2004.

[50] Z. Zhi-Hua and L. Xu-Ying, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 18, pp. 63-77, 2006.

[51] C. J. C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121-167, 1998.

[52] V. L. Berardi and P. G. Zhang, "The effect of misclassification costs on neural network classifiers," *Decision Sciences*, vol. 30, pp. 659-682, 1999.

[53] R. L. Wilson and R. Sharda, "Bankruptcy prediction using neural networks," *Decision Support Systems*, vol. 11, pp. 545-557, 1994.

[54] R. J. Howlett and L. C. Jain, *Radial Basis Function Networks 1: Recent Developments in Theory and Applications*. Heidelberg, New York: Physica-Verlag, 2001.

[55] D. Chakraborty and N. R. Pal, "A neuro-fuzzy scheme for simultaneous feature selection and fuzzy rule-based classification," *Neural Networks, IEEE Transactions on*, vol. 15, pp. 110-123, 2004.

[56] M. Bland, *An Introduction to Medical Statistics*, 3rd ed: Oxford University Press, 2000.

[57] P. Armitage, G. Berry, and J. N. S. Mathews, *Statistical Methods in Medical Research*, Fourth ed. Oxford, United Kingdom: Blackwell Publishing Company, 2002.

[58] J. H. Friedman, "On Bias, Variance, 0/1—Loss, and the Curse-of-Dimensionality," *Data Mining and Knowledge Discovery*, vol. 1, pp. 55-77, 1997.

[59] S. V. Beiden, M. A. Maloof, and R. F. Wagner, "A general model for finite-sample effects in training and testing of competing classifiers," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, pp. 1561-1569, 2003.

[60] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York, NY: John Wiley & Sons Inc, 2001.

[61] L. Breiman, "Rejoinder: Arcing Classifiers," *The Annals of Statistics*, vol. 26, pp. 841-849, 1998.

[62] P. S. Sastry, K. Rajaraman, and S. R. Ranjan, "Learning optimal conjunctive concepts through a team of stochastic automata," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 23, pp. 1175-1184, 1993.

[63] S. Salzberg, "On Comparing Classifiers: A Critique of Current Research and Methods," *Data Mining and Knowledge Discovery*, vol. 1, pp. 1-12, 1999.

[64] S. L. Salzberg, "On Comparing Classifiers: Pitfalls to Avoid and a Recommended Approach," *Data Mining and Knowledge Discovery*, vol. 1, pp. 317-328, 1997.

[65] B. Efron and R. Tibshirani, "Improvements on cross-validation: The .632+ bootstrap method," *Journal of the American Statistical Association*, vol. 92, pp. 548, 1997.

[66] P. Melin and O. Castillo, *Hybrid Intelligent Systems for Pattern Recognition Using Soft Computing*. Berlin Heidelberg: Springer-Verlag, 2005.

[67] R. J. Schalkoff, *Artificial Neural Networks*. New York, USA: McGraw-Hill, 1997.

[68] R. J. Howlett and L. C. Jain, *Radial Basis Function Networks 2: New Advances in Design*. Heidelberg, New York: Physica-Verlag, 2001.

[69] A. J. Washington, *Basic Technical Mathematics With Calculus*, 6th ed. Reading, Massachusetts: Addison-Wesley Publishing Company, 1995.