Modeling Movement Disorders in Parkinson's Disease using Computational Intelligence

Stuart E. Lacy

PH.D.

UNIVERSITY OF YORK
ELECTRONICS

December 2015

Abstract

Parkinson's is the second most common neurodegenerative disease after Alzheimer's Disease and affects 127,000 people in the UK alone. Providing the most appropriate treatment pathway can prove challenging owing to the difficulty in obtaining an accurate diagnosis; due to its similarity in symptoms with other neurodegenerative diseases, it is estimated that in the United Kingdom around 24% of cases are misdiagnosed by general neurologists. A means of providing an accurate and early diagnosis of Parkinson's Disease would thereby enable a more effective management of the disease, increased quality of life for patients, and reduce costs to the healthcare system.

The work described in this thesis details progress towards this goal by modeling movement disorders in the form of positional data recorded from simple movement tasks, building towards a fully objective diagnostic system without requiring any specialist domain knowledge. This is accomplished by modeling established movement disorder markers using Evolutionary Algorithms to train ensembles, before implementing feature design strategies with both Genetic Programming and Echo State Networks.

The findings of this study make an important contribution to the area of data mining, including: the demonstration that Computational Intelligence-based feature design strategies can be competitive to conventional models using features extracted with expert domain knowledge; a thorough survey of evolutionary ensemble research; and the development of a novel evolutionary ensemble approach comprising traditional single objective Evolutionary Algorithms. Furthermore, an extension to a Genetic Programming feature design strategy for periodic time series is detailed, in addition to demonstrating that Echo State Networks can be directly applied to time series classification as a feature design method. This research was carried out in the context of building an applied diagnostic aid and required developing models with means of indicating the most discriminatory aspects of the sequence data, thereby facilitating inference of the precise mechanics of movement disorders to clinical neurologists.

Contents

List of Figures

List of Tables

Acknowledgements

First, I would like to thank Steve Smith and Mic Lones for their support, guidance, and offerings of beer over the last few years. I would also like to thank Andy and Nils, for providing welcome breaks from work and endless PhD discussions; Alan and T, for the deep philosophical lunchtime chats; my examiners Martin Trefzer and Xin Yao for their feedback; Simon, for letting me work on my thesis in the office, otherwise it would never have been completed; my parents, for all their love and care over the years, and never complaining about me being an eternal student; and last but by no means least, I would like to thank Jenny for putting up with my stress, worries, and late nights, and for still choosing to love me at the end of it all.

Declaration

The author declares that all work detailed within this thesis is original, and has not been submitted for any other academic award, at this, or any other institution. References to other published works have been provided where appropriate.

Sections of the work described in this thesis have been previously published in the following journal articles and conference proceedings.

- Lacy, S. E., Lones, M. A., Smith, S. L., Alty, J. E., Jamieson, D., Possin, K. L. & Schuff, N. (2013), Characterisation of movement disorder in Parkinson's disease using evolutionary algorithms, *in* 'Proceeding of the fifteenth annual conference companion on Genetic and evolutionary computation conference companion', ACM, pp. 1479–1486.

- Lacy, S. E., Lones, M. A. & Smith, S. L. (2015*b*), Forming classifier ensembles with multimodal evolutionary algorithms, *in* 'Proc. of the 2015 IEEE Congress on Evolutionary Computation', IEEE, pp. 723–729.

- Lacy, S. E., Lones, M. A. & Smith, S. L. (2015*a*), A comparison of evolved linear and non-linear ensemble vote aggregators, *in* 'Proc. of the 2015 IEEE Congress on Evolutionary Computation', IEEE, pp. 758–763.

- Lones, M. A., Alty, J. E., Lacy, S. E., Jamieson, S., Possin, K. L., Schuff, N. & Smith, S. L. (2013), Evolving classifiers to inform clinical assessment of Parkinson's disease, *in* 'Computational Intelligence in Healthcare and e-health (CICARE), 2013 IEEE Symposium on', IEEE, pp. 76–82.

- Lones, M., Smith, S. L., Alty, J. E., Lacy, S. E., Possin, K. L., Jamieson, D., Tyrrell, A. M. et al. (2014), 'Evolving classifiers to recognize the movement characteristics of Parkinson's disease patients', *Evolutionary Computation, IEEE Transactions on* **18**(4), 559–576.

- Alty, J., Lacy, S., Jamieson, S., Possin, K., Schuff, N., Lones, M. & Smith, S. (2013), 'The sequence effect is not a defining characteristic of Parkinson's disease', *Journal of the Neurological Sciences* **333**, e118–e118.
- Alty, J., Lones, M., Jamieson, S., Possin, K., Schuff, N., Lacy, S. & Smith, S. (2013), 'Clinically 'slight' bradykinesia is accurately detected using a novel device that requires a one-minute test period', *Journal of the Neurological Sciences* **333**, e118–e119.
- Cosgrove, J., Lacy, S., Jamieson, S., Smith, S. & Alty, J. (2015), 'Finger tapping and cognition in Parkinson's', *Journal of Neurology, Neurosurgery & Psychiatry* **86**(11), e4. **URL:** *http://jnnp.bmj.com/content/86/11/e4.83.abstract*

Preface

This work is motivated by the clinical need for an objective and automated means of diagnosing Parkinson's Disease (PD). This requirement is addressed by Computational Intelligence (CI) techniques that take inspiration from nature to offer flexible learning algorithms suitable for a variety of predictive modeling tasks. It is asserted that:

- Current methods of PD diagnosis rely upon subjective interpretations of a variety of information, including movement tasks, which are unreliable even when performed by expert neurologists
- By capturing high resolution positional data, the symptoms of these diseases can be investigated and interpreted in greater detail than is possible with conventional clinical assessment and without any inherent bias
- Owing to their strong search capabilities inherited from biological phenomena, CI techniques are well suited to forming objective means of identifying PD without the assistance of domain knowledge
- Previous approaches into diagnosing neurodegenerative diseases using CI methods have relied upon domain knowledge to guide the development process, or have not considered real world constraints

It is therefore hypothesised that:

> *Computational Intelligence techniques offer an objective and accurate diagnosis of Parkinson's Disease from simple movement tasks measured during conventional clinical assessment.*

Chapter 1

Introduction

Contents

1.1 Parkinson's Disease

Parkinson's Disease (PD) is a debilitating neurodegenerative disease, currently estimated to effect as many as one in five hundred in the UK (*Website of the Parkinson's Disease Society*, 2013). While the cause of the disease is yet to be discovered, it is identified by the gradual deterioration of dopamine carrying neurons in an affected person's brain. This has the effect of causing severe movement disorders—such as bradykinesia, rigidity, and tremor—alongside cognitive impairment. There is currently no cure for PD, only treatment to help deal with the symptoms. However, the most commonly used medication, Levodopa, has a side effect whereby patients who have been taking the drug long term start to display secondary movement disorders, termed *Levodopa-induced dyskinesia* (Keijsers et al., 2003; Manson et al., 2000). Managing the trade-off between these dyskinesias and the original Parkinsonian symptoms can prove extremely challenging and time consuming. As a result, an early and accurate diagnosis is a crucial first step to successful monitoring of the disease in addition to providing appropriate treatment.

Unfortunately, it has been estimated that around 24% of initial PD diagnoses by general

neurologists are inaccurate (Jankovic et al., 2000), due to the similarity in symptoms with other neurodegenerative diseases and physical conditions. An earlier and correct diagnosis would facilitate more appropriate treatment, thereby increasing patient quality of life and reducing the cost to the healthcare system.

Medicine has benefited greatly from advances in transducer technology, as evidenced by the increasing use of non-invasive sensors to capture precise measurements of motion from patients, facilitating the generation of vast amounts of data. Suitable analysis of these data sets can provide a greater level of insight into the underlying mechanisms behind subtle differences in physical movements. The research described in this thesis makes use of movement data recorded from both PD patients and healthy control subjects at clinics around the world, resulting in a large real world data set. Collaboration with medical experts provides motivation and guiding points for a detailed analysis with the aim of producing an increased understanding of not only how movement disorders are manifested as a result of PD, but also how this varies with cognitive impairment.

1.2 Predictive Modeling

Predictive modeling is a field of statistics concerned with predicting the outcome of future events based on past observations, achieved by forming a mathematical model of the data in the form of a mapping of discrete features (the predictors) to an output (the response). This area can be further divided into two categories depending on the nature of the variable being predicted; regression models are concerned with data sets where the response is in the form of a continuous interval or ratio variable, while classification models attempt to predict a discrete nominal variable.

These techniques share similarities, and in some instances the same learning method can be used for both regression and classification problems. Multiple predictive models can be combined into a single system, called an ensemble, to further improve accuracy. This practice mirrors the idea of a committee of experts, whereby knowledge is shared between the members and a decision is reached which takes into account each individual's say. Ensemble classifiers have been successfully applied to pattern recognition tasks, with the underlying principle that each member of the ensemble forms different classification rules from the data by analysing contrasting motifs. Ensembles have been shown to produce more robust predictions and are less susceptible to misclassification errors, provided the members make their errors on distinct

data samples.

Traditional statistical learning algorithms require the input data to be presented to the classifier in a fixed size feature vector with as few elements as possible to provide greater generalising capabilities, in addition to producing a more interpretable model. For sampled movement data this presents problems owing to the high dimensionality of the raw data, and the fact that each pattern can contain a varying number of samples. Thus, a means of reducing the dimensionality into a manageable feature vector is required for efficient modeling.

1.3 Computational Intelligence

Computational Intelligence (CI)—a form of Artificial Intelligence (AI)—describes techniques which enable an algorithm to teach itself to perform a task without requiring expert knowledge of the problem domain. Such techniques are commonly inspired by natural phenomena and include methodologies based on the biological process of evolution, the network of neurons that control a mammalian brain, the behaviour of ant colonies, and others. Applications of computational intelligence include pattern recognition, control, circuit design, and symbolic regression. CI approaches are most advantageous when there is a lack of expert knowledge to apply traditional techniques, or the problem is too mathematically challenging to solve with an analytical approach.

Many CI methods are concerned with optimising computational constructs. These commonly begin by randomly creating a function to attempt the assigned task that will initially perform far below the desired level, but will be improved over time via an iterative teaching process. An advantage of this automated approach to algorithm design is the relative lack of bias involved in the search process. A human programmer would be likely to have a preconceived notion of how to approach the problem, and would thus be limited to a restricted region of the solution space. In contrast, these biologically inspired techniques aim to produce a working solution without regard to how this is achieved. This work focuses on the application of such methodologies to predictive classification problems.

1.4 Contributions

The work described in this thesis has made the following contributions to knowledge:

- The demonstration that feature design strategies implemented using Computational

Intelligence techniques are able to accurately classify PD patients from high resolution time series positional data. Such techniques are shown to perform competitively to conventional modeling of features extracted using domain clinical knowledge. In addition, this technique is demonstrated to offer insight into the relationship between cognitive and motor impairment in Parkinson's Disease patients.

- A thorough survey of evolved ensemble methods from the literature, resulting in a large experiment investigating multiple areas of the ensemble building process, highlighting the trade-off between accuracy and diversity that must be considered when developing ensemble algorithms.

- The development of an evolutionary ensemble learning algorithm using a standard single objective Evolutionary Algorithm (EA), which preserves diversity at the generational breeding stage rather than at the fitness evaluation level. This technique is shown to build ensembles with comparative accuracies to those developed with more complex methods such as combining Multi-Objective Evolutionary Algorithms (MOEAs) and local search methods.

- The extension of a Genetic Programming approach to feature design from periodic time series data by pre-processing the data into its constituent cycles presenting a more natural input method, investigating multiple aggregation functions, and combining the output of multiple trees together in an approach inspired by ensemble classifiers.

- The demonstration that Echo State Networks can be directly employed for time series classification as a feature design method rather than using a kernel-based approach. This technique is attractive for applied modeling, due to the relatively short training time.

1.5 Structure

Chapter 2 discusses neurodegenerative diseases, with a focus on Parkinson's Disease. A review is made of previous studies which have recorded and analysed multimedia data with both standard techniques and predictive modeling approaches. **Chapter 3** introduces the field of predictive modeling and describes current research trends in classification along with an introduction to both ensemble learning and time series classification. **Chapter 4** outlines the history and theory behind the Genetic Programming (GP) and Artificial Neural Network (ANN)

CI algorithms, and details their use for predictive modeling, with a focus on their application to building ensembles and classifying time series data.

An investigation into the suitability of single objective EAs to form ensemble classifiers is described in **Chapter 5**, supplemented with the results on a wide range of benchmark data sets. **Chapter 6** discusses the modeling of PD movement data, including: differentiating between different severities of cognitive impairment, demonstrating the effectiveness of the evolved ensembles to these data sets, and analysing the resultant models to provide a clinical insight into the disease symptoms. An objective feature design method for constructing summary data attributes from raw movement data is introduced in **Chapter 7**, which exploits the flexibility of EAs to classify PD without requiring expert domain knowledge. This technique is investigated in further depth in **Chapter 8**, whereby recurrent Echo State Networks are employed to exploit the temporal element of the position data. **Chapter 9** summarises and concludes, and offers potential directions for extending this work.

Chapter 2

Parkinson's Disease

Contents

2.1 Introduction

The aim of this literature review is twofold: first, to briefly introduce neurodegenerative diseases and how they are exhibited in movement disorders; second, to survey previous efforts at forming computational models of such afflictions. A fundamental outcome of this chapter is the demonstration that as Parkinson's Disease (PD) becomes more prevalent due to increasing life expectancy, an accurate and reliable means of detection will be essential to provide early diagnoses, thereby ensuring maximum treatment efficacy. The survey highlights that previous methods of analysing recorded movement data have involved either investigating an exploratory hypothesis, or attempting to form computational models using features extracted with domain knowledge. Both of these approaches lack a fully objective analysis which, when combined with data mining techniques, can offer a greater insight into the underlying mechanisms of movement disorders.

2.2 Neurodegenerative Diseases

2.2.1 Overview

Neurodegenerative disease is an umbrella term covering a range of medical conditions caused by decreasing neural activity, which typically manifest themselves through movement disorders, in addition to deteriorating neurological functionality. The severity of such conditions are conventionally assessed with questionnaires which take into account both cognitive impairment and any deterioration of movement. The extent of damage to cognitive functionality is measured by inferring how much of an impact the disease has on normal daily life, while the severity of any movement disorders can be assessed using simple movement tasks scored on a numeric scale by a trained clinician. As a result, there is a large amount of data to be collected from both the affected neural activity and precise movement kinematics. Typically, neurodegenerative diseases exhibit themselves in similar, but subtly different fashions; differentiating between conditions can prove challenging.

2.2.2 Parkinson's Disease

Cause

Parkinson's Disease (PD) was first identified by James Parkinson in 1817 (Parkinson, 1817), who described a condition which included tremor at rest, movement impairment, and a gradual degradation over time. It is now known that these movement disorders occur as a result of the deterioration of dopamine producing neurons in the brain. Although there is no commonly accepted cause of the disease, there are several theories, including: the natural human ageing process, a genetic role (Langston, 1989), and an environmental factor—Betarbet et al. (2000) demonstrated that long term exposure to pesticides can bring about Parkinsonian changes to the body. Jenner & Olanow (1998) hypothesised that these potential causes produce oxidative stress which aids neural degeneration. PD is the second most common neurodegenerative condition occurring in adults (Bertram et al., 2005), with incident rates of one in five hundred (*Website of the Parkinson's Disease Society*, 2013).

Symptoms

Due to its similarity with other neurodegenerative conditions, misdiagnosis of PD patients is relatively common; it is estimated general neurologists have a misdiagnosis rate of 24% (Rajput

et al., 1991; Hughes et al., 1992) with this proportion being reduced to 8% for movement disorder experts (Jankovic et al., 2000). However, having movement disorder experts made the initial diagnosis for every incident case of suspected neurodegenerative condition is costly and puts a strain on the healthcare provider. As with many similar diseases, PD manifests itself with several severe motor disorders, including:

- Rest tremor – involuntary small movements when the affected body part is not undertaking any other action.
- Rigidity – the stiffening of muscles leading to motion taking on a cogwheel effect when combined with tremor.
- Akinesia – hesitation in initiating movements.
- Hypokinesia – a decrement in amplitude of movements.
- Bradykinesia – the slowness of movement. Often akinesia and hypokinesia are grouped with slow movements under the umbrella term bradykinesia, but they can be considered separate disorders. When specifically referring to the symptom of reduced movement speed, the term *true bradykinesia* is often used for clarification. This naming convention is used in this thesis.
- Sequence effect – a decrement in amplitude and speed over time when performing repetitive movements.
- Freezing of gait – brief moments where the patient feels unable to move their feet.
- Postural instability – balance issues in patients while standing.

Of these, rest tremor, rigidity, bradykinesia, and occasionally postural instability, are commonly considered to be the cardinal symptoms of the disease (Rajput et al., 1991).

Treatments and Cure

There is currently no cure for PD, however, various treatments are offered to aid in managing the severe symptoms, of which Levodopa (a drug the brain converts into dopamine) is considered to be the gold standard (Brooks, 2008; Poewe et al., 2010). A review led by Schrag & Quinn (2000) discovered that 70% of the surveyed PD patients were being treated with

Levodopa. Unfortunately, Levodopa itself can cause strong side effects, termed Levodopa-induced dyskinesias (LIDs) (Rascol et al., 2000; LeWitt, 2008; Thanvi et al., 2007); these are severe movement disorders with similarities to the bradykinesia caused by the disease itself. The longer the patient spends on Levodopa, the more common LID becomes, eventually progressing to a stage where treatment involves managing the dosage of Levodopa to control the trade-off between the original Parkinsonian symptoms and LID (Rajput et al., 2002).

Assessment

The Unified Parkinson's Disease Rating Scale (UPDRS) (Fahn et al., 1987) is the current gold standard for staging the severity of a patient's disease progression. It is organised into four sections, three of which are self-assessed questionnaires concerning activities of daily living and the patient's motor functionality, while the remaining part comprises a motor examination performed in a clinical setting. Eighteen different tasks are performed and subsequently assessed on a scale of zero to four by the test administrator, with a higher value indicating greater impairment. The score is a compound grade reflecting multiple components of the movement, including speed, amplitude, and rhythm, reducing the complex multi-faceted nature of bradykinesia into a single subjectively measured value. In 2008 the Movement Disorder Society (MDS) sponsored a revision of the UPDRS, addressing various critiques of the original scale, including: rewording questions to reduce ambiguity, changes in wording to highlight differences at the onset of the disease, and the addition of new questions to the questionnaire. The resultant scale was named the Movement Disorder Society sponsored revision of Unified Parkinson's Disease Rating Scale (MDS-UPDRS) (Goetz et al., 2008).

The MDS-UPDRS however, continued to score the motor tests with a single aggregate mark, thereby discarding additional useful information about the patient's movement impairment. To address this limitation, Kishore et al. (2007) developed a new scale to more accurately record all aspects of the movement disorder, called the Modified Bradykinesia Rating Scale (MBRS), which is more sensitive to the different components of movement than the MDS-UPDRS. Recording a more in-depth assessment of the movement disorder is crucial as Heldman et al. (2011) demonstrated that test administrators typically place greater weight on amplitude when marking the MDS-UPDRS motor examination, in addition to studies showing that bradykinesia assessments suffer from the worst reliability out of all the MDS-UPDRS items (Martinez-Martin et al., 1994; Henderson et al., 1991; Bennett et al., 1997; Camicioli et al., 2001). This discrepancy has far-reaching repercussions as Levodopa has been shown to aid movement

speed more significantly than either amplitude or rhythm (Espay et al., 2009). However, while the MBRS offers a deeper insight into a patient's bradykinesia than provided by the MDS-UPDRS, it still lacks a fully objective, quantitative assessment.

In addition to assessing the severity of motor disorders caused by PD, it is crucial to be able to stage any cognitive decline brought on by the condition, as this aspect of PD is becoming increasingly recognised as a symptom. The Montreal Cognitive Assessment (MoCA) procedure focuses on evaluating patient cognitive impairment, rather than the overall perspective provided by the MDS-UPDRS. It was initially derived for assessing cases of Mild Cognitive Impairment (MCI) (Nasreddine et al., 2005), but has subsequently been successfully implemented in PD screening (Dalrymple-Alford et al., 2010; Hoops et al., 2009; Gill et al., 2008). It achieves this by asking various questions assessing the full range of a patient's mental faculties, including visuospatial awareness, memory, attention, and language; contrasting with the MDS-UPDRS section concerning cognitive impact which focuses on the impact on daily living. The entire MoCA can be completed in around ten minutes, adding to its popularity. An alternative means of gauging a patient's cognitive impairment is the Clinical Dementia Rating (CDR). This takes the form of an interview with questions assessing the impact of a patient's cognitive impairment on their daily life, similarly to the MDS-UPDRS. It was initially designed to stage Alzheimer's Disease (AD) but has been commonly used with PD patients too.

Unfortunately, in the United Kingdom (UK) there is currently a deficit of experts on PD, meaning that patients do not always receive the treatment and support that is required to manage such a destructive disease. In addition, access to this limited care varies considerably by region, as discovered from an investigation carried out by the All Party Parliamentary Group for Parkinson's Disease (2009). Therefore, an easy to use and objective means of monitoring the disease would provide significant benefits to patients by facilitating the most effective treatment pathway, and thereby improving quality of life.

2.2.3 Other Conditions

PD is often mis-diagnosed with rarer forms of neurodegenerative diseases, such as progressive supranuclear palsy, multiple system atrophy, and normal essential tremor (Hughes et al., 1992). In addition to these less well known conditions, PD also presents itself with physical and cognitive symptoms similar to those found in more common diseases, such as Alzheimer's Disease and Huntington's Disease. This section briefly summarises these larger diseases to

demonstrate that many of the same challenges encountered with early diagnosis of PD are shared with other conditions, and to reinforce the need for an objective diagnostic aid.

Alzheimer's Disease

AD is the most prevalent neurodegenerative disease as well as the most common form of dementia (Selkoe & Lansbury Jr, 1999). Similar to PD, there is not a single cause of AD, rather a combination of multiple factors including age, environment, genetic factors, and lifestyle are thought to be mainly responsible for contracting the disease (Alzheimer's Society, 2012). Of these influences, age has the most significant impact upon likelihood of developing AD, with prevalence more than doubling between the ages of sixty-five and eighty (Alzheimer's Society, 2012). Symptoms of AD include:

- Memory loss
- Deterioration of eyesight
- Visuospatial problems
- Delusions
- Obsessive behaviour

Currently there is no cure for AD (Alzheimer's Association, 2012; Alzheimer's Society, 2012), however, drugs have been developed to help slow the progress of the disease and alleviate some symptoms (Nordqvist, 2009). The most important treatment for AD is to have care provided for the patient, particularly as their condition deteriorates, however, this can prove costly and extremely emotionally difficult.

Assessment of AD generally includes physical examinations as well as cognitive tests. The mental assessments are typically concerned with the patient's reasoning and memory faculties and can be scored in various ways, with one example being the Rey-Osterrieth Complex Figure (ROCF) (Rey, 1941; Osterrieth, 1944). The ROCF is a drawn figure comprising numerous shapes combined together in a seemingly random manner; the patient must first copy the drawing, and then a short time later reproduce it from memory. This exercise assesses the patient's visuospatial ability as well as their memory. As a form of dementia, AD has a greater impact upon an individual's cognitive functionality than PD, with less of a deterioration to motor facilities.

Huntington's Disease

Huntington's Disease (HD) is a hereditary condition—whereby a person has a 50% chance of developing the disorder if a parent is a gene carrier—which presents itself in numerous ways including cognitive, motor, and psychiatric symptoms. HD patients can appear to have movement disorders similar to those found in PD, in addition to random uncontrollable jerky movements called chorea, rigidity, and postural problems. Cognitive impairments include memory issues—which can progress into dementia—in addition to depression, anxiety, aggression, and obsessive behaviour. As with many other neurodegenerative diseases, there is no cure, however, drugs exist to manage the symptoms. In the case of HD, medication developed for PD patients has been shown to help deal with chorea. Standard psychiatric medication can also be used to treat some of the other symptoms, such as depression, irritability, and mood swings.

Similar to how the MDS-UPDRS rates a patient's PD progression, there is a standardised rating scale for HD called the Unified Huntington's Disease Rating Scale (UHDRS). As with the MDS-UPDRS, it assesses both physical and mental impairments over four areas: motor, cognitive, behavioural, and functional. Overall, HD differs from PD in that it presents greater cognitive impairment, although PD medication has been used successfully to treat motor disorders in HD patients.

2.3 Studies into Movement Disorders

This section surveys studies which have analysed data recorded from PD patients, primarily in the form of positional data although other sources are discussed in brief. PD patient data has been used in both experimental trials to investigate a hypothesis (described in Section 2.3.1), and analysed using predictive modeling techniques to form complex computational models of Parkinsonian movement disorders (summarised in Section 2.3.2).

2.3.1 Investigative Experiments

Kinematic Studies

As technology has progressed, medical studies have increasingly used recorded data in order to gain a deeper and objective understanding of the underlying mechanisms of diseases and illnesses. In particular, various studies have investigated Parkinsonian movement disorders by

obtaining positional data.

Before transducer technology progressed to the point of wearable sensors, such studies used invasive equipment such as the apparatus used by Benecke et al. (1987) and Agostino et al. (1992). Benecke et al. (1987) demonstrated that PD patients were particularly unable to accurately switch from one motor movement to another sequentially, in comparison to healthy controls. The intrusive apparatus involved measuring elbow and finger position during motion; a potentiometer and a strain gauge were used to measure kinematic features, whilst electromyography signals of the body part were recorded to gain insight into the underlying physiology. Similarly, Agostino et al. (1992) looked at sequential arm movements in PD, HD and Dystonia. The patient traced patterns whilst holding a device on which two potentiometers were attached. They found that sequential movements were abnormal in all three diseases and in long duration sequences the slowness of movement is particularly noticeable in PD. Both cumbersome and intrusive, these tests performed in a clinical setting were distinct from the patient's home environment where the movement disorders would generally occur and consisted of performing unfamiliar movement tasks.

Since then, sensors have become progressively smaller—and therefore less invasive—resulting in numerous benefits for study participants, including a reduced awareness of the recording equipment and less discomfort, helping to provide a more relaxing test environment. Many studies have used positional and rotational sensors to provide accurate measurement of the affected body part's position in space. Agostino et al. (2003) developed previous methods (Agostino et al., 1998) for analysis and used optoelectronic sensors to obtain 3D results of both individual finger tapping motions and that of four fingers tapping on a thumb. Taking into consideration the number of finger taps, size, pauses and time taken, they found that patients with PD had more severe motor disorders during individual finger taps as opposed to whole hand taps. This study is notable for its use of a standardised movement task—finger tapping—to provide a familiar environment for the participant and also ensuring the results are replicable.

Additional studies measuring various movement features are listed in Table 2.1.

Table 2.1: Kinematic Studies of Parkinson's Disease (PD)

Authors	Date	Focus
Manson et al.	2000	Ambulatory dyskinesia monitors
Berardelli et al.	2001	Effect of sensory cues on bradykinesia
Hoff et al.	2001	Objectively assessing LIDs
Bonato et al.	2004	Measuring longitudinal motor fluctuations
Iansek et al.	2006	Improving freezing of gait with external cues
Chee et al.	2009	Reducing step length of freezing of gait
Patel et al.	2009	The use of quantifiable data to estimate severity of symptoms
Rodrigues et al.	2009	The slowing of successive finger tap motions
Yokoe et al.	2009	Different features of finger tapping
Espay et al.	2011	Quantifying how bradykinesia responds to dopaminergic medication
Heldman et al.	2011	Examining the MBRS reliability with quantifiable measures
Ling et al.	2012	Reducing hypokinesia in PD patients and Progressive Supranuclear Palsy (PSP)

These studies demonstrate that there is a focus in the neurodegenerative research community on examining the effects of motor disorders brought on by these conditions, by taking advantage of progress in transducer technology to record highly precise movement data. In all of these studies, a standard feature extraction and statistical analysis was employed to inspect the data, thus requiring expert domain knowledge alongside an exploratory hypothesis. The field of data mining, however, typically involves an unbiased analysis of data to identify significant relationships and patterns. Such techniques can be applied to PD to provide a more detailed understanding of the precise kinematics of the disease.

Non-movement PD studies

Data has also been recorded from other, non-movement sources, in PD studies. For example, various imaging modalities have been employed to analyse the underlying neural activity of PD patients. Other novel tasks have been developed to examine the impact of the disease upon various functions, such as identifying tremor in a shape tracing task (Aly et al., 2007), quantifying the impact of PD treatment on the sequence effect and 'true' bradykinesia using a simple pegboard test (Kang et al., 2010), and even comparing the speech patterns of PD patients with healthy controls by identifying vocal disorders when pronouncing vowel sounds (Tsanas et al., 2012).

2.3.2 Diagnostic Modeling

Alongside improving hardware transducer technology, computer software technology has progressed to the stage where learning algorithms can be trained on vast data sets in reasonable time frames, resulting in complex computational models of neurodegenerative diseases (more detail on these algorithms is provided in Chapter 4). For example, Keijsers et al. (2003) used Artificial Neural Networks (ANNs) to model ninety-two movement features extracted from PD patients while performing activities of daily living in order to assess the severity of LID. The features were processed from six accelerometers attached to the subject's body at different locations, with fifteen summary measures calculated for each sensor location for each minute window of data, with an additional two features resulting from the proportion of time the patient was sitting and standing. The features typically comprised summaries of the processed separation signal, both in the time and frequency domain, utilising *a priori* knowledge of the frequency bands where dyskinesia typically are located (1-3Hz). While the study achieved strong results, albeit with a small sample size of thirteen patients, it had a contrasting aim to that presented in this work, where diagnosing PD is the goal rather than identifying LID. While not using movement data, Ericsson et al. (2005) investigated a PD diagnosis system using Single-Photon Emission Computed Tomography (SPECT) imaging. In particular, eighty nine patients had their basal ganglia imaged using SPECT, from which seventeen summary data attributes were extracted using expert knowledge, before being input into Support Vector Machines (SVMs) for classification. While an interesting approach, imaging tests can prove expensive and potentially discomforting for the patients themselves, in contrast to movement tasks.

Patel et al. (2009, 2010) analysed dyskinesias and bradykinesia by modeling movement data from six UPDRS tasks, with the models attempting to predict the severity score assigned by a clinician. The data was recorded by eight accelerometers attached to a patient's limbs, before being pre-processed using prior domain knowledge of frequency bands of interest. For example, features of tremor were identified in the 3-8Hz band, while bradykinesia and dyskinesias are located in frequencies less than 3Hz. In total, seven features comprising Digital Signal Processing (DSP) measures were extracted from each window of the data, since each test was split into thirty equally sized segments; the number of windows to use was a parameter to be optimised. As stated, the goal was not diagnosis, but rather predicting the severity score as measured by the UPDRS. To this end, models were trained to each individual rather than a global approach, furthermore the sample size was limited at only twelve patients.

Modeling at the individual rather than population level removes the ability to form inference about the causes of movement disorders, and necessitates a pre-processing and modeling pipeline for every additional patient. The features themselves were input into a SVM for severity classification.

While all the above studies produced accurate models and provide value as practical diagnostic aids, they all required expert domain knowledge to extract summary features from the raw data to be subsequently input into standard predictive model learning algorithms (more detail of this field can be found in Chapter 3). However, only one study—that by Patel et al. (2009, 2010)—has focused on diagnosing PD from movement data, yet this is achieved at the individual level by modeling severity of movement periods in contrast to diagnosing subjects as having the disease or not. Not only does this methodology prevent overall diagnosis of new subjects, but it inhibits inference about the underlying movement disorders. Furthermore, similarly to the work of Keijsers et al. (2003), these approaches are not flexible to new problems, such as distinguishing between different neurodegenerative diseases, due to their reliance upon domain expertise to form the feature vectors. In addition, the samples from both of these studies are somewhat limited by both their sizes, and their lack of geographic variety amongst the patients. To better establish the ability of classifiers to accurately predict PD, a large sample comprising patients from multiple locations and demographic backgrounds is required.

Previous work by Lones, Smith, Alty, Lacy, Possin, Jamieson, Tyrrell et al. (2014) assessed a single cohort of PD patient and control on finger tapping data to classify the patients, using Computational Intelligence (CI) techniques that are described in greater detail in Sections 4.4.4. The work described in this thesis extends this research from both an algorithmic perspective and the application domain. A collaboration with PD experts provided the motivation for an investigation of the suitability of the UPDRS finger tapping task duration (as detailed in Appendix B); facilitating the extraction of discrete summary features representing bradykinesia from the positional data. A primary motivation of this work is comparing CI classifiers operating on the raw data with models of these summary features. A larger data set was available for this work than that used by Lones, Smith, Alty, Lacy, Possin, Jamieson, Tyrrell et al. (2014), allowing for a greater insight into the complexities present when modeling real world data recorded at different locations where subtle differences in the assessment methodology might be present. In addition, the work detailed in this thesis describes a preliminary investigation into modeling cognitive impairment of PD patients, a

more challenging application than distinguishing between a person having the disease or not.

Algorithmic extensions include the development of an ensemble classifier learning algorithm using Evolutionary Algorithms (EAs), producing a model of the extracted summary features suitable for the relatively small sample size of the PD finger tapping data set–as described in Chapter 6. Section 7.1 details extensions to the Genetic Programming (GP) windowing technique introduced by Lones, Smith, Alty, Lacy, Possin, Jamieson, Tyrrell et al. (2014) to produce more suitable feature design strategies for this data. Furthermore, a novel application of reservoir computing for PD diagnosis is investigated in Chapter 8.

2.4 Conclusions

As average life expectancy increases, a corresponding rise in the prevalence of neurodegenerative diseases is observed, bringing with it a large number of issues associated with the high cost of treatment. Due to their closely related nature, several of these diseases present with visually similar motor disorders, including Parkinson's Disease, Alzheimer's Disease, and Huntington's Disease; causing relatively high misdiagnosis rates. Several investigative experiments into PD have been performed using position sensors to provide an accurate and objective overview of the related motor disorders.

There have also been a limited number of previous approaches into applying predictive modeling to Parkinsonian movement disorders, typically employing expert domain knowledge of the frequency bands of interest to extract summary features from the raw data. As detailed in Section 2.3.2, these studies have often been focused on identifying LID rather than diagnosing PD, or differentiating between neurodegenerative diseases. While such approaches have resulted in accurate models, the requirement of clinical guidance for the feature design process limits the applicability to new sources of movement data or application areas, including disease identification. An objective method for analysing movement data would thereby provide numerous benefits, including adaptability to new data sources. Previous studies have also been hindered by relatively small sample sizes, thereby limiting the ability to form inference about the underlying disease characteristics. Ultimately, to be implemented in a real world application, any reliable form of disease detection needs to overcome several limitations with medical data acquisition, including small sample sizes and noise.

Chapter 3

Predictive Modeling

Contents

3.1 Introduction

Predictive Modeling is a technique used to build mathematical models of expected future behaviour, formed by analysing past events. It can be broadly divided into two categories based on the type of the predicted value; if the problem involves forecasting a continuous value then a *regression* model is employed, *classification* models are used when the response takes the form of a categorical variable. The work discussed in this thesis solely involves classification problems, typically *binary* problems where the two dichotomous classes represent a person having a disease or not. This chapter provides an overview of the statistical concepts built upon in later chapters, including: the fundamentals of classification models, considerations to

be made when developing models for real world problems, ensemble classification strategies, and classifying time series.

3.2 Classification

3.2.1 Overview

Classifiers are models which attempt to predict a nominal or ordinal response variable from a finite set of known possibilities. The prediction is based on known information about the data instance, typically in the form of a fixed-size vector containing summary features of the sample, as either continuous or nominal measures. More formally, classifiers form a mapping between an input feature vector $\mathbf{f} = [f_1, f_2, \ldots, f_n]$ and an output prediction p_i.

The output of a classification model, p_i, can take numerous forms depending on the architecture of the model being used and the characteristics of the data set being analysed; the most straightforward approach is to output a discrete class prediction as one of the classes in the set of possible classes $\{c_1, c_2, c_3, \ldots, c_C\}$. The advantage of this technique is that it does not require any further interpretation and is thus advantageous for simple situations where estimates of certainty in the prediction are not required. However, a data instance which is borderline between two classes will have the identical output to a clearly separable pattern.

Certain learning algorithms expand upon this and output a measure of confidence for their vote. Such systems typically output a vector of continuous values $[o_1, o_2, o_3, \ldots, o_C]$, for each input pattern, where each value corresponds to a measure of support for that particular class. These values can take the form of a measure of probability such that $\sum_{i=1}^{C} o_i = 1$, or can be arbitrary units indicating a general score. In cases where the model outputs a score for each possible label, the overall predicted class can be established as the one with the maximum score, and the confidence measure can be converted to share the characteristics of a probability estimate by using the softmax transformation (Bridle, 1990).

In certain cases—typically when predicting dichotomous binary variables—models output a single continuous value rather than one for each class. A label vote can be produced by thresholding the score with respect to either an arbitrarily chosen level, or a pre-determined value. Such approaches present advantages for medical applications where the aim is to predict the existence of a disease, allowing the model score to be interpreted as an additional biomarker to aid diagnosis.

3.2.2 Evaluating Classification Models

Overview

Clearly, a means of comparing multiple classifiers needs to be established. The simplest method is to calculate the ratio of correctly predicted data objects, shown in Equation 3.1 for a dichotomous response, where A_c and B_c are the number of correctly predicted instances of classes A and B respectively, and N represents the total number of patterns.

$$Accuracy = \frac{A_c + B_c}{N} \tag{3.1}$$

This value is known as the classifier's *accuracy*, however, it has several limitations when employed as a measure of goodness of fit. First, if the classes are imbalanced in the data set, then the classifier could predict every pattern as belonging to the modal class resulting in a high accuracy. To illustrate this, if a data set consisted of ninety items belonging to class A, and ten from class B, a classifier predicting that every sample belongs to class A would achieve 90% accuracy, yet it is far from a useful classifier as it only achieves the *no-information rate*. While this is an extreme example, it highlights the bias towards modal classes and the necessity of taking the class distribution into consideration when evaluating models.

Another major concern with using accuracy to rate classifiers, is that it offers no distinction between Type I and Type II errors made by the model. Generally classifiers such as these are developed to aid in real world decision making, where there is a different penalty associated with each error type. For example, in medical diagnosis there is arguably a difference between misdiagnosing a healthy person, over giving an affected person the all clear. Accuracy, as a measure of classification ability, does not allow this issue to be considered. It is for these reasons (see Provost et al. (1998) for a detailed discussion of these flaws) that Receiver Operating Characteristics (ROC) have become widely used for evaluating the performance of classifiers in the machine learning community.

Receiver Operating Characteristics

Receiver Operating Characteristics (ROC) were developed in World War II to aid radar engineers by providing a more detailed analysis of detection rates. They have since become widely accepted by the biomedical community in particular (Zweig & Campbell, 1993; Pepe, 2000), and are well established in other domains including machine learning (Bradley, 1997).

		Prediction	
		Positive	Negative
Actual	Positive	True Positive	False Negative
	Negative	False Positive	True Negative

Table 3.1: ROC confusion matrices display the possible outcomes of a binary prediction

The principal idea behind ROC is to identify samples according to their predicted and actual outcome, rather than simply marking predictions as correct or incorrect as with the accuracy measure. Referencing its original use, the two possible response classes are referred to as *positive* and *negative*, indicating the presence and absence of an object respectively; ROC are also commonly used in medical applications where the same naming conventions are appropriate to signify the existence or absence of a condition. However, ROC can be applied to any dichotomous binary data set regardless of what the response variable represents. To calculate ROCs, a contingency table is formed with the four possible predicted/actual outcomes (as shown in Table 3.1), providing a thorough overview of a classifier's performance. A correct prediction of a positive class is thereby labelled True Positive (TP), while a correct prediction of a negative class is separately labelled True Negative (TN). Similarly, errors are identified as either False Positive (FP) or False Negative (FN).

From the ROC confusion matrix two useful summary measures can be derived. The first, *sensitivity*, is the ratio of positive classes which were correctly predicted (Equation 3.2), also known as the True Positive Rate (TPR). Likewise, the *specificity* of a classifier is the proportion of negative classes which were accurately guessed (Equation 3.3) and is hence termed the True Negative Rate (TNR). An additional attribute is the False Positive Rate (FPR), which is defined as 1 - specificity, and measures the ratio of false positives to all negative classes (Equation 3.4).

$$\text{TPR} = \frac{TP}{TP + FN} \tag{3.2}$$

$$\text{TNR} = \frac{TN}{TN + FP} \tag{3.3}$$

$$\text{FPR} = \frac{FP}{TN + FP} \tag{3.4}$$

ROC utilise the TPR and FPR to represent the performance of a classifier with respect to the

two different error types, resulting in a (FPR, TPR) pair summarising the model fit, as opposed to the single quantifiable measure of accuracy. The majority of classification models have at least one tuneable hyper-parameter, allowing for multiple (FPR, TPR) pairs to be calculated, one for each operating point. By varying the hyper-parameter value, the trade-off between the sensitivity and the specificity can be managed, allowing the user to select a final model with the desired error characteristics. An alternative means of obtaining a series of (TPR, FPR) data points is to use a classification model with a single output score, and calculate the sensitivity and specificity at each new threshold score of the positive class, as seen in Table 3.2—this method is analogous to the manner in which biomarkers are evaluated for their diagnostic ability in medical research. Plotting these (TPR, FPR) data pairs as connected lines then produces a ROC curve as shown in Figure 3.1.

Table 3.2: Calculating TPR and FPR values from a model outputting a score

Score	**Class Label**	**TPR**	**FPR**
0.82	1	0.33	0.00
0.71	1	0.67	0.00
0.67	0	0.67	0.25
0.52	0	0.67	0.50
0.41	1	1.00	0.50
0.35	0	1.00	0.75
0.20	0	1.00	1.00

ROC curve diagram

The ROC curve depicts the TPR plotted against the FPR, allowing the user to visually inspect the trade-off between sensitivity and specificity to select the desired operating point, in effect providing multiple candidate classifiers from a single model. An ideal classifier—that is to say one that accurately predicts every example—will have a TPR of 1 and FPR equal to 0 at every threshold. On the ROC curve this is expressed as two straight lines connecting (0,0) to (0,1) and then to (1,1). A random guess would lie on the line $TPR = FPR$, and is shown visually in Figure 3.1 as a dashed line. Inspecting the plot allows for an in-depth assessment of the classifier's performance profile, however in certain circumstances a single quantifiable measure is required, particularly when comparing multiple models across several data sets.

Area Under Curve

A popular method of obtaining a single measure of a model's fit is to calculate the area under the ROC curve, thereby producing an unbiased measure of the classifier's accuracy

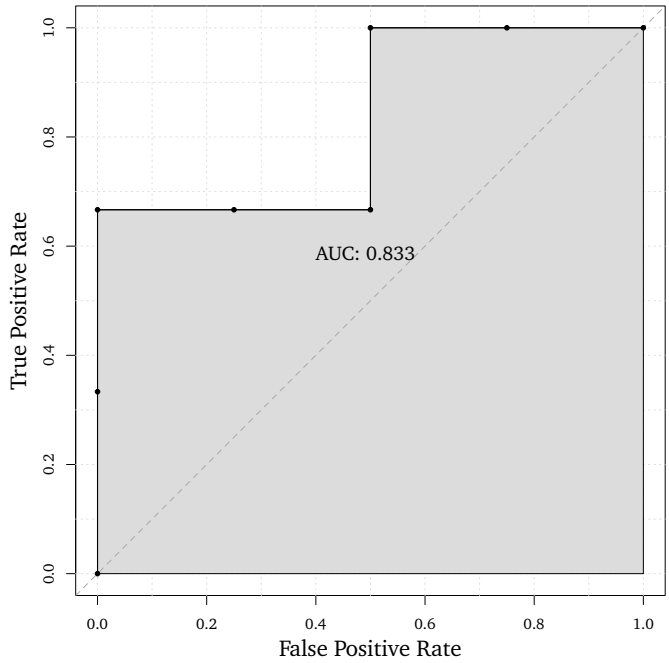


Figure 3.1: The (TPR, FPR) pairs from Table 3.2 can be plotted as a ROC curve

over all thresholds. Formally, the Area Under Curve (AUC) is the probability that a randomly chosen data instance from class 1 will be assigned a higher score than a randomly chosen member belonging to class 0. An AUC of 1.0 indicates a perfect classifier, while a classifier scoring 0.0 mislabels every prediction; an AUC value of 0.5 indicates that the model has the same expected discriminatory ability as a random guess. A classifier with an AUC of less than 0.5 therefore is ranking the class 0 patterns higher than those from class 1. This is still evidence of discriminatory ability however, and a useful classifier can be formed by inverting the predictions. For example, a model with an original AUC of 0.3 has an effective value of 0.7.

While the AUC has been widely adopted in many fields—including machine learning—due to its objective nature (Bradley, 1997), its use has also come under some criticism due to alleged faults with interpretation (Hand, 2009; Lobo et al., 2008). Lobo et al. (2008) take issue with the fact that the AUC value summarises the classifier's performance over areas of ROC space in which one would never normally operate (for example areas of extreme specificity and sensitivity). Furthermore, while visual inspection of a ROC curve allows for selecting the trade-off between FP and FN errors, the AUC does not and weights them equally. This is appropriate for situations where a robust model needs to be developed to be used over a variety of operating conditions, but is not relevant when the misclassification costs are

known prior to forming the model and will not vary. Hand (2009) cites the major problem concerning the use of the AUC to compare classifiers being the fact that the misclassification penalty for the same error type is different for two independent classifiers. He presents an alternative measure, called the H-measure, to solve this issue. However, as with the issue of benchmarks in machine learning research, it is never possible to get the full research community to support a single method. While the AUC is not a perfect evaluation measure, it is superior to other commonly used alternatives—such as accuracy—and for these reasons is utilised as the primary means of model evaluation throughout this work.

3.2.3 Overview of Learning Algorithms

There is a large range of learning algorithms which fit classification models to training data, all approaching the problem from different perspectives. Broadly speaking, these techniques can be grouped into several categories: linear models, non-linear separators, rule based approaches (including trees), and other. This section only discusses some of the more well known and commonly used methods from each of these groups.

Logistic Regression

Logistic Regression (Cox, 1958) is arguably the simplest form of classification model. It extends linear regression analysis to model problems with a categorical dependent variable by outputting the log odds of a positive class—termed an *event*. This is achieved similarly to its regression counterpart, and thereby can be grouped into the General Linear Model (GLM) category. A linear expression between the covariates and the log odds is formed, where the coefficients are fitted using maximum likelihood estimation. The coefficients represent the increase in the log odds of an event for a one unit increase in the corresponding predictor, enabling detailed insight into how the prediction is being formed. It is typically only applicable for binary problems, although extensions have been made for the multinomial case. A similar technique is Linear Discriminant Analysis (LDA), which also attempts to form a linear model by decreasing the probability of misclassification, however this will not be discussed further in this thesis.

Decision Trees

Rule-based models are formed using a series of *if then... else...* conditions. For example, a model predicting whether a person will pass an exam or not based on their work habits could look as follows:

> ***IF*** *wake up time is 10:00 or earlier* ***AND*** *studying time is 8 hours a day* ***THEN*** *the person will pass the exam*

Models can take the form of a linear series of such rules, classifying the sample into the first group that is successfully met, such as the PART algorithm by Frank & Witten (1998). However, a more common approach is to build a hierarchy of rules into a (typically) binary tree with the branches from each node representing the *IF* and *ELSE* paths. Such models are known as decision trees, or classification trees in the specific case when being used to predict a categorical value. Following a path for a given data instance down the tree finishes at a leaf, which represents the predicted class label.

The two best known implementations of decision trees are Classification and Regression Trees (CART) by Breiman et al. (1984) and C4.5 which was developed by Quinlan (1993). These two algorithms produce similar models but differ slightly in the fitting process. An updated version of the C4.5 learning algorithm—named C5.0—has been developed which incorporates the boosting ensemble strategy to further improve model accuracy along with other modifications (Quinlan, 2004). An advantage of decision trees is that they are very interpretable, detailing the threshold predictor values which separate the data. Furthermore, they provide an inherent means of feature selection, as only the data attributes deemed to be discriminatory by the induction process are included in the final model. This requires less pre-processing on the part of the practitioner, which can sometimes introduce bias into the procedure. In addition, classification trees can support categorical predictors, by splitting the possible outcomes into two subsets, for each branch. A drawback of decision trees is their tendency to overfit to the training data, caused by allowing very specific rules in deep trees which only identify small fractions of the training data.

k-Nearest Neighbours

The k-Nearest Neighbours (kNN) algorithm predicts the class of an unknown pattern based on its similarity to previously seen samples from the training set of which the class is known.

Specifically, it calculates the distance to every point in the training data using a distance metric (typically Euclidean distance) of the feature vector. The k training samples with the smallest distance are identified (*nearest-neighbours*), from which a majority vote of the labels determines the class prediction for the unknown data point. It can prove sensitive to the choice of value for k, and does not tend to work well with high dimensionality data sets. In addition, modeling using kNNs requires storing the training data, which can prove problematic for large data sets, and the resultant classifiers have limited interpretability.

Support Vector Machines

Typically, learning algorithms attempt to produce a function which separates every training sample from each class, with little consideration for exactly how this is achieved. For example, for linearly separable data, there are an infinite number of discriminatory lines which can be produced to split the data into the respective classes. Support Vector Machines (SVMs) take a different approach to traditional classification algorithms which produce any dividing line using all the training data, and instead attempt to produce the optimal boundary by using only the samples from each class closest to the perimeter, i.e. the hardest to classify points.

The optimal boundary is fitted to the training set according to the samples closest to the margin—known as the *support vectors*—with any future data being compared against this border to form a prediction. The actual values of the support vectors' feature scores are not used directly during classification, instead unknown samples have their similarity to the competing support vectors calculated with the label being assigned to the most similar class. Similarity between two samples is typically calculated using the dot product—resulting in a linear classification boundary—however one of the strengths of the SVM learning algorithm is that this similarity function can be replaced by any *kernel function* to produce highly non-linear boundaries. This is achieved with the *kernel trick*, which maps the two comparison feature vectors to a higher dimension where they are more easily separable, and obtaining their similarity in this new plane.

There are several commonly used kernel functions, including:

- Linear
- Radial basis function
- Polynomial

SVMs have been extremely popular owing to their flexibility in terms of kernel function, and high accuracy rates—they are frequently the gold standard classification algorithm on many data sets (Cortes & Vapnik, 1995). However, there is a large element of trial-and-error in the design of a SVM model due to the range of candidate kernel functions, each containing hyper-parameters. In addition, the resultant classifiers can prove challenging to interpret owing to the importance placed on a small number of training points—the support vectors.

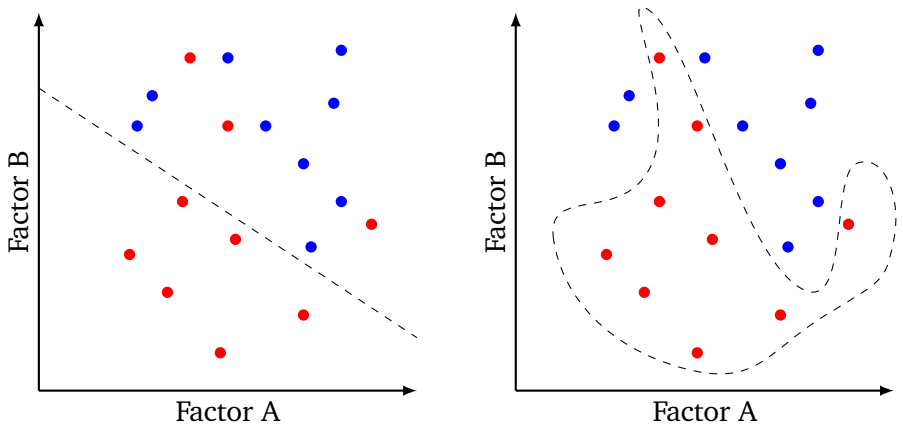
3.3 Managing Overfitting

The most significant consideration when developing predictive models is how well they will adapt to unseen data sets. If a classifier can accurately identify patterns in a given training set, but not correctly predict any new data, then it is of little practical value. This phenomenon is termed *overfitting* and occurs when the model has learnt the specific patterns evident in the training data rather than any overall relationships evident in the wider population.

A simple and effective method of measuring overfitting is to partition the data set into a training and test set, a typical setup involves the test set being half the size of the training set. The model learns from the training set samples, then has its unbiased performance assessed on the previously unseen test set. If it can also achieve a high accuracy on this unseen data, then it can be said to generalise well. However, simply randomly allocating the data into training and test sets can introduce an additional element of bias into the results, as there is no guarantee the test set is representative of the population data. One method of reducing this selection bias is to employ *k-fold cross-validation*. This comprises partitioning the data set into k equally sized *folds*, iterating through each fold using it as the validation set and the $k-1$ remaining folds as the training samples; and then averaging the validation fold AUC scores. This allows for a more thorough assessment of a model which has been fit on different splits of the data set.

To observe why overfitting occurs, the *bias-variance decomposition of error* can be used. This is a derivation—originally developed for linear regression (Geman et al., 1992)—which states that the error of any model (initially framed as Mean Square Error (MSE)) comprises the sum of three distinct values, the *irreducible error*, which is unavoidable noise in the data set; a *bias* term stemming from the overall fit of the model to the data; and error due to *variance* which results from how adaptable the model is to variation in the predictor values. Figure 3.2 shows two different models classifying a binary data set with two continuous predictors, where there

is a clear non-linearly separable relationship between the explanatory variables and the class. The decision boundary shown in Figure 3.2a contains bias error, but will adapt well to unseen data, while the classification function shown in Figure 3.2b has learnt the small discrepancies in the training set, producing a model with extremely low bias error but will not generalise well to new samples.



(a) A model with high bias error (b) A model with high variance error

Figure 3.2: Comparison of two models' decision functions (shown as dashed lines) with contrasting error characteristics

All models contain noise from both of these sources, but learning algorithms differ in the proportion of the two minimisable terms. For example, decision trees typically contain low bias error as they can match the training set very well. However, when predicting unseen samples they are susceptible to subtle changes in the predictor values from the training set, resulting in large values of variance error. The k parameter of the kNN learning algorithm can control this trade-off, low values of k produces models with high variance and low bias error, but as k increases the bias term becomes the greater source of error. SVMs can manage the trade-off by tweaking the cost parameter c, which dictates the size of the margin of the boundary between the two classes. A smaller value of c indicates that the cost associated with misclassifying training points is not so significant and allows the margin to be larger, reducing error due to variance at the expense of bias. Thus, to produce well generalised classification models, the variance term needs to be minimised.

Typically, a model with low bias and high variance can be improved more easily than the other way around, as small increases in bias error can greatly improve the generalisation capability of the classifier. While this can usually be achieved by tweaking a learning algorithm parameter, a novel approach that has become extremely popular in the machine learning

community is to build an *ensemble classifier*. This consists of a collection of models typically having low bias error but large variance. By combining the outputs from the base classifiers, any misclassifications made by one—which would normally go unnoticed—will be overruled by the remainder of the ensemble, provided they do not also make an error on the same pattern. The following section describes ensemble classifiers in further detail.

3.4 Ensemble Learning

3.4.1 Overview

Ensembles are collections of classifiers that when combined provide greater accuracy than any of the individual members. Figure 3.3 shows the standard method of forming an ensemble, called *overproduce-and-select* (Giacinto et al., 2000; Kuncheva, 2003), and highlights the numerous considerations to be made when developing an ensemble system.

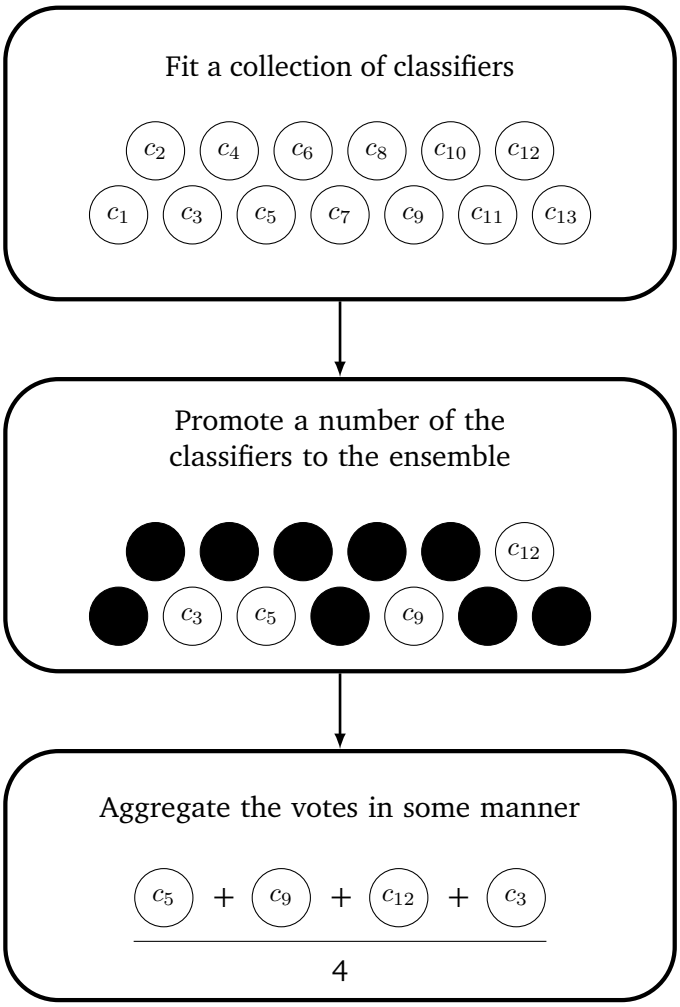


Figure 3.3: The stages of overproduce-and-select

A major aspect of ensemble building comprises making decisions concerning the manner in which the base classifiers are trained, which can include models differing in the choice of learning algorithm, parameters, or training data samples. Once the pool of classifiers has been collated, a method of selecting which ones to promote to the ensemble must be determined. This could vary from using all of them to only a small ratio, with a large number of possible selection criteria. Finally, once the ensemble has been formed, a voting aggregation strategy needs to be applied to produce the overall prediction; again this can be achieved in a multitude of ways.

Classifier ensembles have become increasingly popular in machine learning research owing to their ability to offer greater accuracy and robustness than a single model. Dietterich (2000) presents three explanations for the recent success of ensembles.

1. Statistical: The ensemble can average its members' votes and reduce the risk of selecting the wrong class.

2. Computational: Ensembles are typically created from running a local search from many starting points, thus increasing the global search possibilities.

3. Representational: By taking weighted sums of the members' votes as the ensemble's output, the space of representable functions is increased.

Hansen & Salamon (1990) argue that a required condition for an ensemble to have greater predictive power than its members is that the individual classifiers are accurate and yet diverse in their predictions. While the advantages of a diverse ensemble are clear, namely that different members misclassify different patterns and will be outvoted by the majority, Brown et al. (2005) demonstrate that there is little consensus in the machine learning community on exactly what constitutes diversity, and cites the lack of a formal definition as a drawback. The following section details different methods by which diversity can be maintained in an ensemble of classifiers.

3.4.2 Methods of Preserving Diversity

There are numerous stages during the ensemble development process where diversity can be introduced or preserved, to optimise ensemble efficacy.

Base Classifier Models

An effective means of producing a varied pool of base classifiers is to employ an assortment of learning algorithms. As evidenced from the *no free lunch* theorem (Wolpert & Macready, 1997), there is no single best classification model for every possible data set; by increasing the range of learning algorithms used in the ensemble, the likelihood of finding a strong model increases. Furthermore, each architecture will map the features to an output differently, resulting in varied classification rules.

Learning Algorithm Hyper-Parameters

Many learning algorithms have at least one tuneable hyper-parameter, for example k in kNN models, the value of c in SVMs, and the minimum number of observations per split in decision trees. By using variants of the same model formed using different hyper-parameter values, the ensemble diversity can be increased.

Training Set Manipulation

One of the most commonly implemented diversity preservation methods involves training the base classifiers on different data samples. Having the base models fitted on different subsets of the training set enables them to identify distinct patterns and trends to use as the basis for their predictions. The most straightforward method of manipulating the training set is to use cross-validation, whereby the training data is divided equally into k folds. The model is fitted on $k-1$ of these subsets, leaving one out for validation at each iteration. This training and evaluating process is repeated k times to produce k classifiers which have been trained on subtly different data sets. A similar approach is to use bootstrap samples, where new training sets are formed by a process of sampling with replacement from the original training set. Any data instances not selected are deemed the *out of bag* samples and can be used to validate the model.

Data Feature Manipulation

A similar approach to the training set manipulation described above, is to form base models on varying subsets of the feature space, having the same result of classifiers identifying differing patterns in the training data. The simplest such approach would be to fit each model using

only one attribute, but distinct or overlapping subsets can also be used. This technique is most effective when applied to data sets where each feature contains useful information, rather than those with a large feature vector but with many redundant attributes. Along with the training set manipulation diversity preservation mechanic, this technique is one of the most frequently employed approaches for ensemble building.

Modify Output Classes

For multi-class problems an alternative approach to promoting diverse classifiers is to divide up the data set according to the output classes. This is the idea behind Dietterich & Bakiri's (1995) error-correcting output coding technique, whereby the original classes are randomly subset into two groups, with the data appropriately relabelled. The learning algorithm is then presented the new data set as a standard binary classification task. This sub-setting and subsequent training is repeated numerous times with a new classifier formed at each iteration which can distinguish between the two randomly selected groups. An ensemble is subsequently formed from the classifiers produced at each step of the resampling loop. An alternative approach would resample the data set into two, consisting of one class and every other instance, and training a classifier. Repeating this for every class would form models that specialise in identifying a single class.

Selection Process

The final stage of the ensemble building process where diversity can be preserved is in the member selection phase. Given a large pool of base classifiers, there are numerous ways to choose which models will be used by the ensemble. For example, the base classifiers could be selected primarily based on their individual accuracy, or an explicit measure of diversity could be implemented. The size of the ensemble has an impact upon its performance as well; an ensemble consisting of three members will have a smaller range of opinions than one with twenty base classifiers.

3.4.3 Voting Strategies

Once the base classifiers have been trained and selected for the ensemble, the final stage of the building process is to decide on a voting strategy to produce the overall prediction for a new data pattern. This largely depends on whether the base classifiers output label votes,

continuous scores, or probabilities. To facilitate more interpretable voting schemes, scores and probabilities can be converted into a hard vote as discussed in Section 3.2.1, however, using continuous values allows for a wider range of voting methodologies.

Discrete Labels

The most popular vote combination strategies for ensembles of models predicting class labels are:

- Majority Vote (MV): The label predictions are tallied with the ensemble prediction being the class with the highest number of votes. This is the simplest method of voting and has proven to be effective.
- Weighted Majority Vote (WMV): Weight the base classifiers' votes by a coefficient—often related to their accuracy—and then take a majority vote. In this paradigm the MV technique can be viewed as an instance of WMV with weights set to 1.

The advantage of using discrete labels is that it allows for simple interpretation of the voting procedure and results in a straightforward prediction. However, it offers no degree of support or confidence in the votes; a data instance that is borderline between two classes receives the same prediction as one that is far from the separation boundary.

Continuous Output

Predictive models which output a continuous value can provide a hard class label upon the application of a threshold for a single output case, or the maximum value where the classifier outputs a support measure for each class. This addition of a confidence measure provides greater information to the ensemble upon which to base the overall prediction. Several methods for aggregating continuous outputs have been used in the literature, including:

- Average: The widely used equivalent of a MV for continuous values is to calculate the mean base classifier output.
- Weighted average: As with discrete labels, the ensemble members can be weighted before having their mean value calculated.
- Other algebraic functions can be used to aggregate the members' votes, such as Maximum, Minimum, Sum and Product.

- By regarding the continuous output as a measure of support for a class, fuzzy voting methods have been implemented, including Decision Templates (Kuncheva et al., 2001).
- Instead of a simple combination function, the base classifiers' outputs can be input into a secondary classifier, such as an Artificial Neural Network (ANN) to produce the ensemble output.

3.4.4 Training Ensembles

An additional aspect to be considered when developing an ensemble is whether to train it or not. Training an ensemble involves optimising the constructed combination model to more accurately represent the data, however, care must be taken to not overfit to the training set in doing so. Duin (2002) discusses the benefits of this supplementary facet of ensemble building, along with describing considerations to make when following this route.

This additional training step can take numerous forms. One approach is to use the standard overproduce-and-select paradigm to produce a number of base classifiers, but then optimise their combination with a training algorithm, rather than just applying a voting strategy. Alternatively, rather than using a pool of statically fitted models, the base classifiers could be trained with their later inclusion in an ensemble in mind. This allows for greater ensemble accuracy at the cost of a more intensive training process.

3.4.5 Example Approaches

This section provides a brief overview of the most commonly used ensemble building techniques.

Bagging

Bagging, developed by Breiman (1996*a*), aims to create a diverse collection of classifiers by iteratively running the same learning algorithm on different subsets of the original training set. These bootstrap replicates are selected at random with replacement, producing dissimilar versions of the data which can contain multiple copies of data samples. Once a set number of models have been trained the process stops and the ensemble is formed from all the classifiers. To predict future data patterns, the outputs of all the members are combined with the MV technique. The name *bagging* derives from the two stage process of **B**ootstrap **AGG**regat**ING**.

Bagging is particularly effective on unstable predictors (such as decision trees), which have high variance error characteristics.

Boosting

Similar to bagging, *boosting*—developed by Valiant (1984); Kearns & Valiant (1994)—produces a diverse classifier by manipulating the training set to focus models on different samples. This is achieved by taking a learning algorithm and training a model on a random subset—chosen without replacement in contrast to bagging—of the data set. The data instances are weighted according to their classification accuracy, so that the second model to be fitted will be rewarded more highly if it manages to correctly predict the instances that the first model failed on. In this manner, the criteria of having an ensemble of classifiers who make their errors on disjoint parts of the data set is explicitly managed. After each iteration, the weights are updated to reflect the current ensemble's performance.

Once a predetermined number of iterations has completed, the ensemble is formed and can be used to predict new patterns by aggregating all the members' votes. AdaBoost (Freund et al., 1996, 1999) is the most well known algorithm in this field although numerous similar techniques exist under the name *gradient boosting*. It uses a WMV aggregation scheme, with the weights derived from the base classifiers' performance on the training set. While boosting can be employed with any learning algorithm, it is frequently used to strong effect with decision trees; Breiman (1996*b*) explains the reasons for this success. Referring back to the bias-variance decomposition discussed in Section 3.3, boosting (as with many ensemble techniques including bagging) works most effectively by reducing the variance of low-bias classifiers (Johnson & Rayens, 2007).

Due to boosting's similarity to bagging, the two techniques have often been compared. AdaBoost has demonstrated higher classification accuracy than bagging (Freund et al., 1996), although in test problems with noise artificially added AdaBoost suffers more severely than bagging (Dietterich, 2000).

Random Forests

Random Forests (Breiman, 2001) extend the bagging concept to focus on one specific base classifier learning algorithm, namely decision trees. A large number—typically in the order of a magnitude of thousands—of decision trees are trained similarly to bagging, in addition

to a feature subsetting method to ensure that the base classifiers form their predictions on a variety of features. The use of two diversity preservation techniques helps to ensure a robust and accurate ensemble is built, with far better generalising capabilities than that typically encountered with decision trees, which have a tendency to overfit.

3.5 Classifying Time Series Data

Traditional predictive modeling comprises mapping a discrete set of summary data attributes to an output class prediction. The data features can take the form of continuous or nominal values, but typically the order in which they are input into the classifier has no bearing on the resultant model. With the recording of data of all forms of media being on the rise, the issue of how to classify sequenced data is becoming increasingly common. Xing et al. (2010) developed a taxonomy of sequence classification approaches with three categories—model-based, distance-based, and feature-based. Model-based techniques attempt to generate mathematical models of the underlying phenomena, and then form predictions based on aspects of the model behaviour. Distance-based techniques compare the similarities between two sequences, this can be implemented neatly within the framework of a SVM using a tailor-made kernel function for the input data or with a nearest neighbour method. The final approach to classifying sequence data—including time series—is to summarise the raw data into discrete features. This is typically achieved using domain knowledge to identify discriminatory facets of the sequence.

Currently, the gold standard time-series classification techniques are distance based; calculating the pairwise similarity between the sample under assessment and every pattern in the training set and forming the prediction using a 1-nearest neighbour (1NN) model. Two of the most common similarity functions are the Euclidean distance and Dynamic Time Warping (DTW), which is similar to the Euclidean distance but can also take into account if the signals are out of phase. Bagnall et al. (2016) surveyed the literature of time-series classification methods and found that despite recent developments 1NN classification combined with DTW is still the most accurate method overall. Similar results have been found by other studies, including: Xi et al. (2006), who attempted to fix the primary limitation of DTW of being computationally expensive; Ratanamahatana & Keogh (2004), who also investigated means of improving the DTW algorithm in terms of both accuracy and speed; and Jeong et al. (2011), who incorporated weights into the algorithm.

Of the three time-series classification approaches, both Harvey & Todd (2014) and Bagnall et al. (2012) identify the feature-based technique as having the greatest potential for forming accurate classification models. Bellman (1961) first derived the phrase "the curse of dimensionality" to refer to the large increase in problems encountered in data with greater dimensionality. While Bellman was working in the area of dynamic optimisation, Duda et al. (1973) argue that his theory has implications for pattern recognition, in addition to other fields. This highlights the importance of reducing the dimensionality of time series data into a format more easily modeled by a classifier.

As mentioned previously, this process—termed *feature design*—typically necessitates *a priori* domain knowledge to reduce the set of potential summary features into those where there is likely to be some discriminatory power. There have been several attempts at automating the feature design process, which will be briefly summarised now. DTW has been employed with feature design, for instance by Rodríguez & Alonso (2004) who incorporated decision trees to classify features by comparing input patterns to reference series at each decision node. Geurts (2001) used feature design to classify speech signals, by forming naïve summary features (such as the average) of segments of time series, in addition to extracting features with regression trees. These features were then modeled by standard classification algorithms such as decision trees and 1NN. Deng et al. (2013) developed an algorithm called Time Series Forest (TSF), which works similarly to Random Forests by having a large number of trees—termed *time-series trees*—attempting to identify discrete aspects of each pattern. The trees provide a prediction by splitting the input time series into intervals, and comparing the results of two intervals being input into one of three functions at each node, with the three functions comprising the mean, the standard deviation, and the slope of a regression line. The training process attempts to form these splits to be as discriminatory as possible. As with random forests however, the models formed by this technique contain a large number of constituent trees, and still contain a large number of features which may require further reduction before further modeling. Fulcher et al. (2013) developed a framework to visualise differences in time series from various applications comprising over nine thousand summary measures, including statistical summaries, signal processing algorithms, and information theory approaches. While this technique provides a useful method for comparing sequences from different sources, it does not enable a simple classification, as for any application the feature set would need to undergo further dimensionality reduction.

3.6 Conclusions

Statistical predictive modeling contains numerous techniques for predicting outcomes based on prior data, making them extremely suitable to aid the development of an automated diagnosis of Parkinson's Disease (PD). Ensemble classifiers in particular have been shown to perform well with relatively small sample sizes, and can build models which achieve higher accuracy than an individual classifier by reducing the error due to variance at the cost of a slight increase in bias error. They have significant potential in predicting information about a person based purely on movement characteristics recorded by accurate position sensors. The most effective approach of classifying time series data is to compare similarity between sequences using DTW, or employ feature design techniques to reduce the dimensionality into discrete manageable feature vectors. However, DTW is computationally expensive and does not offer any significant insight into which regions of the sequence are most discriminatory. Furthermore, there is no standard feature design approach, and using *a priori* knowledge is favoured when available. The following chapters will discuss means of using Computational Intelligence (CI) for improving two aspects of predictive modeling: building accurate ensemble classifiers, and extracting features from time series—thereby removing the need for domain knowledge.

Chapter 4

Computational Intelligence

Contents

4.1 Introduction

Computational Intelligence (CI) is a collective term for a category of optimisation algorithms and representations including: Genetic Programming (GP), Artificial Neural Networks (ANNs), Particle Swarm Optimisation (PSO), Ant Colony Optimisation (ACO), Fuzzy

Logic, and Artificial Immune System (AIS). A common characteristic of CI techniques is that they are designed to solve complex problems with little awareness of the underlying data, typically taking inspiration from naturally occurring phenomena. The two main implementations used in this work are GP and ANNs. This chapter introduces both of these algorithms, details their development history along with practical implementation considerations, discusses how CI can be applied to predictive modeling with a focus on ensemble systems and time series classification, and finally surveys the literature for previous work utilising this combination.

4.2 Genetic Programming

4.2.1 Overview

Genetic Programming (GP) is a field of CI research which takes inspiration from the natural biological process of evolution to automatically generate functioning computer programs. In particular, Evolutionary Algorithms (EAs) are used as global optimisers to iteratively train a population of candidate programs encoded by primitive data strings called genotypes, with the behaviour being governed by phenotypes—typically represented as standard computing data structures comprising operators acting on inputs. Training is carried out by an iterative process of selecting high performing individuals and combining them to form new candidate solutions, thereby extending the search at each step. In the context of predictive modeling, the model itself takes the form of a complex (often non-linear) expression which is stored in a data structure such as a syntax tree, graph, or list; rather than fitting a model to the data using maximum-likelihood estimation, an explicit user specified *fitness* criteria guides the evolutionary search.

4.2.2 History

Turing (1948) first proposed the idea of a randomly created network in his discussion of *unorganised machines*, which could be taught to perform specific tasks via a “genetical search”. While these ideas were developed in the context of binary networks to mimic the human brain, there are strong similarities between the fundamental behaviour of unorganised machines and EAs. EAs themselves were first developed by Barricelli et al. (1954) as an optimisation algorithm, however, they remained relatively underused for several decades until John Holland’s work on Genetic Algorithms (GAs) (Holland, 1973, 1975), which used an EA to evolve

a string of numbers, representing chromosomes; EAs and GAs subsequently obtained success in optimising parameters for complex tasks. Cramer (1985) produced the first work on what would later become GP, whereby the chromosome evolved by a GA encoded a structure based on a tree; Koza (1990, 1992, 1994) expanded on this idea and remains extremely influential in the field. He demonstrated the successful application of GP to a variety of problems as well as developing key techniques. With the increasing computational power available in the modern era, GP has been applied to more complex problems than ever before; the advent of large multi-core Graphics Processing Units (GPUs) and supercomputer clusters has allowed for increasingly distributed techniques which are well suited to the population based search provided by EAs.

4.2.3 Implementation

This section provides an overview of the practical considerations when applying GP to a problem by detailing the development and procedure of tree-based GP, chosen as it is the most common representation (others are detailed in Section 4.2.4).

Program Representation

GP combines the iterative population-based training procedure of an EA with programs acting on the specified inputs using a collection of primitive functions. There are two aspects to each evolved individual, the genotype and phenotype. The genotype takes the form of a string of primitive data types—typically binary or integer values—which dictate the corresponding individual's functional representation. The phenotype, in contrast, governs an individual's behaviour by representing the program in the form of a standard computational data structure. The traditional phenotypical representation of an evolved expression is a syntax tree, with branch nodes describing a particular function and leaf nodes acting as either inputs to the program, or numerical constants. GP uses domain specific functions, for example, an expression evolved with GP attempting to solve a symbolic regression problem would implement mathematical operators, while a program controlling a robot would include more appropriate functions such as "move north" or "turn around". By simply adapting the function set used by the tree, functions can be evolved unique to each application. This is in contrast to another commonly used CI technique, Artificial Neural Networks, where every node commonly shares the same function regardless of the problem. The use of a genotype facilitates simple modification of the genetic code, however, in order to run the encoded program, an interpreter

is required to parse the specified function. Figure 4.1 shows an example arithmetic syntax tree, for which a genotype could be written as the constituent functions in pre-order notation: $+(-(5,2),\times(3,3))$. To convert this into a string of primitive values to be used as a chromosome, the functions need to representable by basic data types, a look-up table provides a simple means of achieving this. An example chromosome for the tree shown in Figure 4.1 could be: **98 96 5 2 99 3 3**, whereby high value numbers are used to differentiate between program inputs and functions.

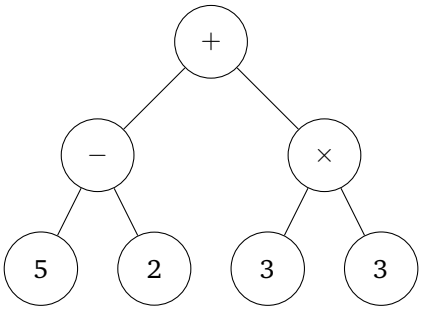


Figure 4.1: A syntax tree representing the expression $(5-2)+(3\times 3)$

Evolutionary Algorithms

EAs are used in GP to search the solution space of potential programs to find the optimal configuration for the given problem. The core component of survival of the fittest—continuous mixing of genes with favourable traits surviving—is adapted as the basis for EAs. Darwin (1859) first proposed the theory that every species of wildlife currently existing has reached its current state via a continuous process of adapting favourable traits from ancestors. Individuals that inherited characteristics that were more likely to enable them to survive would thereby be more likely to pass on their genes; this concept is known as "survival of the fittest". As understanding of the human genome increased, this led to a more gene centric perspective of evolution, popularised by Dawkins (2006). This view states that genes compete for survival, with alleles more likely to aid in survival being passed on to future generation.

Evolutionary genetics are adapted as the underlying mechanism behind the EA search algorithm and this influence is reflected in the choice of terminology employed by the research community. The algorithm maintains a collection of candidate solutions, known as a *population*, which is iteratively ameliorated over a series of *generations*. The primitive string representation of a candidate solution (its genotype) is also termed a *chromosome* with each component referred to as a *gene*. At each generation, every individual in the population has its phenotype formed and evaluated for the given set of inputs, requiring the use of an

interpreter to convert the chromosome into its corresponding syntax tree and execute the expression. Each candidate solution is then assigned a score (its *fitness*) representing its success at completing the specified task, analogous to a species' survival rate in nature. For classification tasks this is typically the accuracy or Area Under Curve (AUC), however, as with the choice of function set, the fitness function is application-dependent. For example, in the case of a population being evolved to control a robot to complete several tasks, the fitness value could represent the time taken by the candidate solution to complete the tasks, where the objective is to minimise this value.

At each generation after every individual has had its fitness calculated, a predetermined number of candidate solutions are selected from the population—typically according to their fitness—to be *parents* for the next generation, this is known as the *selection* phase. Child solutions are derived from the parents via various *breeding* methods, which tweak the parent's genotype to form new genetic material. In natural evolution breeding provides a means for successful genes to be passed down to future offspring, it likewise has an important role in EAs as it allows for the continuation of the optimisation search to new parts of the solution space via two operators, *crossover* and *mutation*. Crossover entails combining the genotype from two parents into a single one for the child, while mutation modifies alleles at random. Crossover typically helps the search escape local optima and increases the amount of exploration, while mutation acts as more of a local search. However, both of these properties can be tweaked using various hyper-parameters such as the *mutation rate*, and the ratio of crossover to mutation. Figure 4.2 shows the individual from Figure 4.1 after undergoing mutation, and an example of crossover is displayed in Figure 4.3. Child solutions are either automatically inserted into the next generation, or compete with their parents for survival, depending on the choice of *replacement* operator.

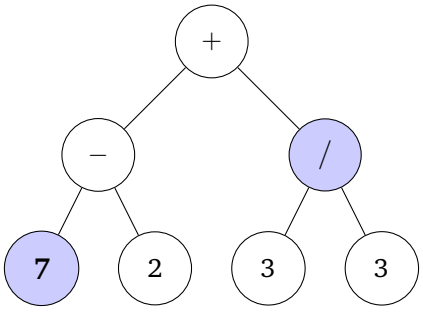


Figure 4.2: The tree from Figure 4.1 after undergoing mutation, now representing the expression $(7 - 2) + (3 \div 3)$

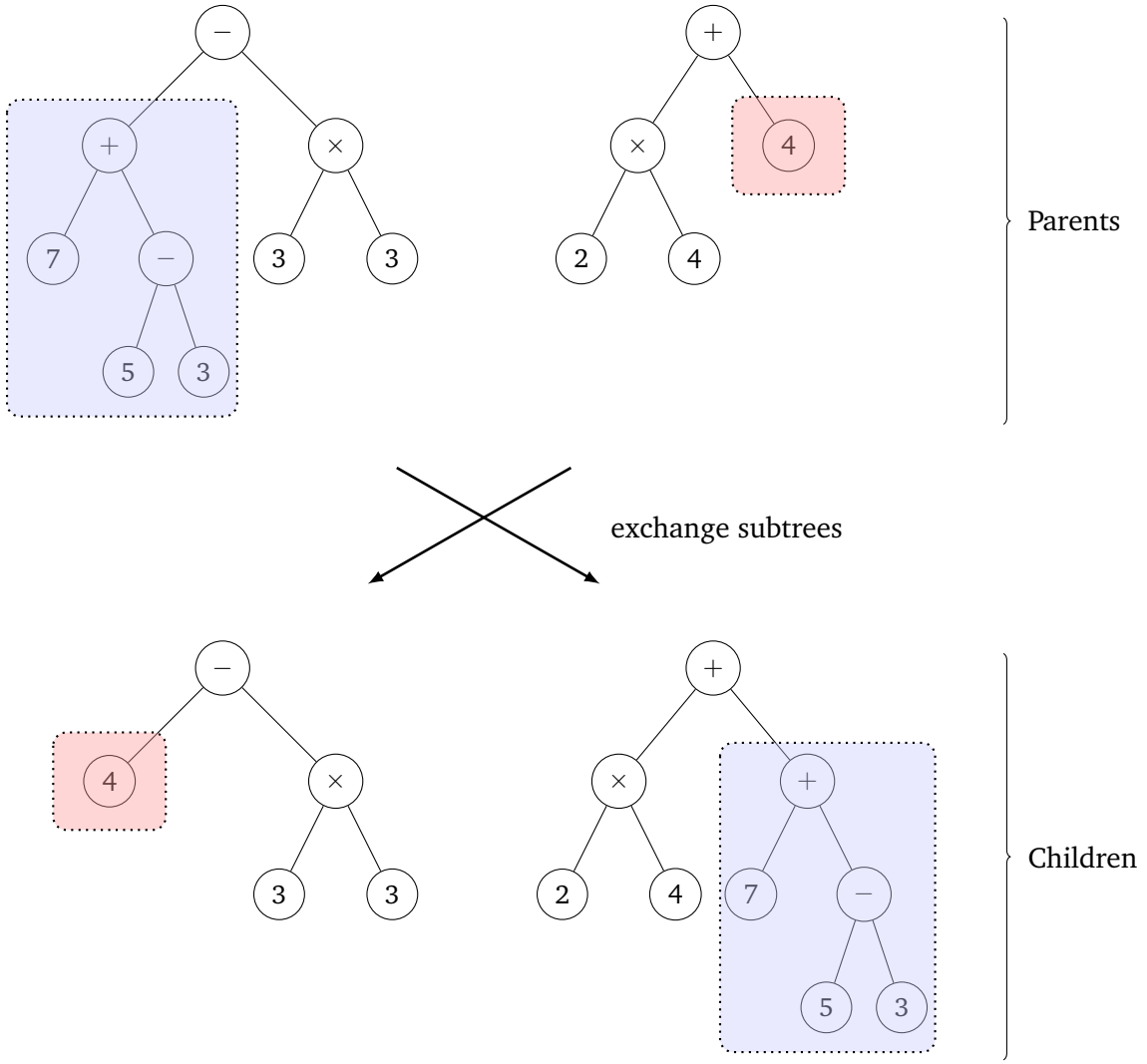


Figure 4.3: Children share genetic material from both parents when formed via crossover

This process of evaluation, selection, breeding, and replacement, repeats for the predetermined number of generations, with the fittest individual in the final generation selected as the overall winner and is the output of the algorithm.

Owing to its flexibility—as both the fitness function and the operators available to each candidate are domain specific—GP has found success in numerous fields, including temporal forecasting, symbolic regression, signal processing, classification, and control tasks. An additional contributory factor to the success of EAs is their ability to combine local and global searches to produce a thorough canvas of the solution space. Numerous considerations in the development of an EA allow for the tweaking of the *exploration* vs *exploitation* dynamic. Explorative searches are those closest to a random search, they attempt to cover a large area of the solution space without concentrating on smaller regions. Exploitative searches, on the other hand, act more similarly to hill climbing, typically using configurations which allow

for only small modifications to the existing generation, providing thorough searches in small regions to identity any maxima. Manipulation of this trade-off can produce very different searches; most applications implement a balanced approach.

4.2.4 Variants

Since Koza's (1990) work on tree-based GP, numerous other phenotype representations have appeared in the Evolutionary Computation (EC) literature. One such encoding comprises a sequence of low level commands with little communication between them, data is instead shared through registers. This technique—termed Linear GP—removes the need for parsing complex data structures but at the cost of interpretability of the evolved expression. The most common implementation of Linear GP represents the commands sequentially in a list (Brameier & Banzhaf, 2007), although stack based versions have also been implemented (Perkis, 1994). Representing the program as a graph—as opposed to a tree—has also been researched in the form of Parallel Distributed GP (Poli et al., 1997) and Cartesian Genetic Programming (CGP) (Miller & Thomson, 2000). The use of a graph structure allows for the reuse of certain nodes, which in tree based GP is achieved through duplicating a sub-tree; an example CGP phenotype is displayed in Figure 4.4.

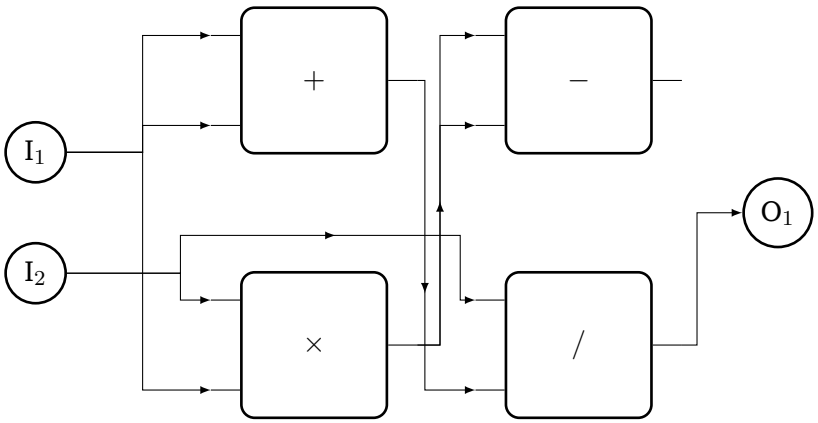


Figure 4.4: CGP programs are represented by graphs arranged in Cartesian grids

Alongside differing program representations, there are a variety of adaptations to the standard EA template offering a vast range of search behaviours. There are numerous areas where behaviour can be modified, including:

- Overall hyper-parameters including the population size and generation limit
- Selection method; both deterministic and probabilistic techniques have been used to

select parents with and without replacement

- The choice of breeding operators, along with hyper-parameters governing the amount of modification performed at each generation

- The manner in which individuals are selected for the subsequent generation by the replacement operator

The effect of manipulating these areas is often to address the previously discussed exploration/exploitation trade-off. For example, the Evolutionary Strategy (ES) is an EA variant which provides a much more exploitative search by means of incorporating low population sizes, combined with a large number of iterations and generally lack of crossover.

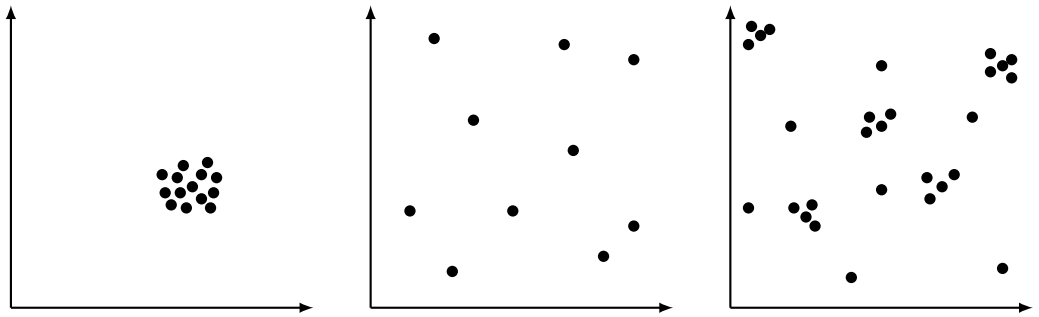
Standard EAs optimise a single fitness criterion in their search process, however, for some applications there are multiple competing objectives to be improved; an EA evolving the design of a circuit layout may attempt to build the circuit that produces the desired behaviour most closely, but also one which minimises the amount of raw materials used. In such situations Multi-Objective Evolutionary Algorithms (MOEAs) are ideally suited. These techniques allow for a fitness function comprising more than one guiding measure, whereby candidate solutions are evaluated using the concept of Pareto dominance. Two popular MOEA strategies are NSGA-II (Deb et al., 2002) and SPEA2 (Zitzler et al., 2001). MOEAs can also prove beneficial for multi-modal problems, when *a priori* knowledge indicates that there are multiple areas of the solution space where favourable solutions exist. Multi-modal problems do not necessarily require the optimisation of multiple fitness criteria however, and standard EAs can be applied to such problems with minor modifications (detailed further in Section 4.2.5).

Distributing the evolution of individuals has also become widely researched. Such approaches aim to emulate natural evolution by partitioning the global population of individuals into sub-populations, which are subsequently evolved independently; genetic characteristics can be shared between the sub-populations by permitting the exchange of individuals. The increase in parallel computing techniques have facilitated the growth of such techniques, however it has been shown that this is not necessary for most applications (Poli et al., 1999). Distributed evolution has even been achieved on a global scale, using the internet as a global link (Chong & Langdon, 1999; Draves, 2006; Klein & Spector, 2007; Langdon, 2005).

4.2.5 Niching

Overview

Standard EAs form a global search of the solution space in an effort to optimise a single value, the fitness function, thereby converging on a point by the end of the evolutionary process. For some applications, however, *a priori* knowledge indicates that there are multiple beneficial areas of the fitness landscape to explore, rather than a single optimum. In such cases a multi-modal search is required. *Niching* algorithms extend base EA functionality to preserve subsets of the population exploring different areas of the solution space (known as *niches*), providing a multi-modal search. Figure 4.5 visualises the difference between the standard exploitative EA search, an explorative (i.e. random) search, and that provided by niching algorithms on a 2D solution space. There are numerous algorithms that fall under the niching umbrella, however, most achieve a similar result by manipulation of the selection and replacement phases of the EA cycle.



(a) Exploitative (b) Explorative (c) Niching

Figure 4.5: Niching algorithms attempt to search around multiple focal points in the solution space, rather than converging on a single point

Deterministic Crowding

Crowding is a form of niching first derived by De Jong (1975), which adapts the selection and replacement portions of a standard EA to promote diverse individuals by forcing offspring to compete against their most similar parents for promotion into the next generation, thereby preserving useful niches. During breeding, a pair of individuals are randomly selected and bred together to produce two offspring candidate solutions, which are subsequently paired against their most similar parent in a competition for survival. The most common form of crowding, termed Deterministic Crowding (DC) (Mahfoud, 1992, 1995), holds a direct tournament with the most accurate of the two candidate solutions gaining promotion to the

subsequent generation. This process allows for useful niches to be discovered in the population and preserved for future generations by more accurate offspring, increasing both the accuracy and the breadth of the search at the same time.

In the original implementation a structural measure of similarity was employed, for example the Hamming distance could be used to assess two bit string chromosomes' similarity. However, its effectiveness as a diversity measure is limited; the child solutions are only assessed for similarity against their parents rather than the whole population, and structural similarity does not necessarily correlate to behavioural similarity. For example, a parent may be searching in a useful area of the solution space, but is replaced by a child which, while sharing some genetic code, does not operate in the same niche of the search process.

Probabilistic Crowding

Probabilistic Crowding (PC) is a variant of DC developed by Mengshoel & Goldberg (1999), which aims to generate a more explorative search of the solution space by holding a probabilistic tournament for survival rather than a deterministic one. The selection and replacement process is carried out in the same manner as for DC until the tournament between the parents and their most similar offspring is held; under PC, the probability that a parent p is promoted over its most similar child c is a function of their respective fitnesses (from fitness function $f(x)$), and is shown in Equation 4.1. This results in a more diverse population than one evolved via DC, albeit one which can contain considerably less fit individuals as survival is no longer directly related to goodness.

$$p(p) = \frac{f(p)}{f(p) + f(c)} \tag{4.1}$$

Restricted Tournament Selection

Restricted Tournament Selection (RTS), developed by (Harik, 1995), has a similar outcome to DC albeit achieved through slightly different means. As with both of the crowding implementations, two parents are selected at random from the population and bred together to form two offspring. Each of these child solutions are subsequently paired against the most similar individual from a subset of the new population—rather than their most similar parent as is the case with crowding—with the number of competing solutions in the subset determined by the hyper-parameter w (*window size*). This tournament is of a deterministic fashion with the fittest

individual gaining entry into the population, ensuring that each niche in the population is represented by its strongest individual. Tweaking w can change the behaviour of the algorithm, a smaller value increases the chance of the child surviving, but at the expense of potentially limiting the number of niches in the population.

Species Conserving Genetic Algorithm

Species Conserving Genetic Algorithms (SCGAs) were introduced by Li et al. (2002) as an alternative multi-modal niching algorithm, which explicitly attempts to preserve multiple sub-populations during a GA run. The metaphor of biological evolution is extended to account for competing species in the global population, whereby each species represents a niche in the EA solution space. Species are defined as the set of individuals within a certain degree of similarity of the species' *dominating individual*, the threshold of which is known as the *species distance*—represented as σ_s. Before breeding occurs, the species are determined by locating the dominating individuals through an iterative process of establishing the candidate solutions which do not lie within the species distance of any existing species, and using these to form new species. The next generation is subsequently produced following the standard rules of selection, crossover, and mutation. Afterwards, any of the original species dominating individuals which were not selected for the new generation are copied in and hence 'preserved'. The choice of σ_s controls the number of species—and thereby niches—in the population.

4.3 Artificial Neural Networks

4.3.1 Overview

Artificial Neural Networks (ANNs) are computational models of the intricately connected system of billions of neurons which comprise human brains, and are used as computational tools in addition to finding use as an aid to study the brain and its functionality. Their main feature is a massively distributed network of individually weak computational units—neurons—which, when combined, are able to generate large amounts of computational power (Haykin, 1994). They were developed with the intention of producing a computational resource with the same processing power as humans, albeit via simplifying the model to facilitate easier comprehension and implementation. By applying appropriate learning algorithms, these networks can adapt to solve a variety problems without requiring underlying knowledge of the

problem area. By taking inspiration from biology, it is hoped that these techniques can tackle challenges that standard computers struggle with, but the human brain excels at, pattern recognition for example. While ANNs have been researched with the aim of generating an accurate and biologically rigorous model, in this thesis they are only considered in the context of CI.

4.3.2 History

Initially, the development of ANNs grew out of a desire to develop an accurate model of the brain in order to further our understanding of how it functions. The first breakthrough in ANN research was achieved by McCulloch & Pitts (1943) who developed the first mathematical model of a neuron, shown in Figure 4.6. They considered a neuron to consist of weighted inputs and a binary output, which is set as high when the sum of the weighted inputs is greater than a specific threshold. Hebb (1949) introduced his hypothesis of 'Hebbian learning', which states that the synaptic weight between two neurons increases with connection usage. The next milestone in the field was the development of the *perceptron* by Rosenblatt (1958), which combined the McCulloch-Pitts model with Hebb's idea of dynamic adaptive synaptic weights, and introduced a bias term appended to the sum of the weighted inputs. He also worked on the idea of supervised learning, and developed an algorithm for modifying the synaptic weights for use in binary classification problems. Throughout the 1960s, Widrow and Hoff made several developments, including: the Least Mean Squares (LMS) algorithm for updating weights and the Adaptive Linear Neuron (ADALINE), which combined the McCulloch-Pitts neuron with LMS (Widrow et al., 1960).

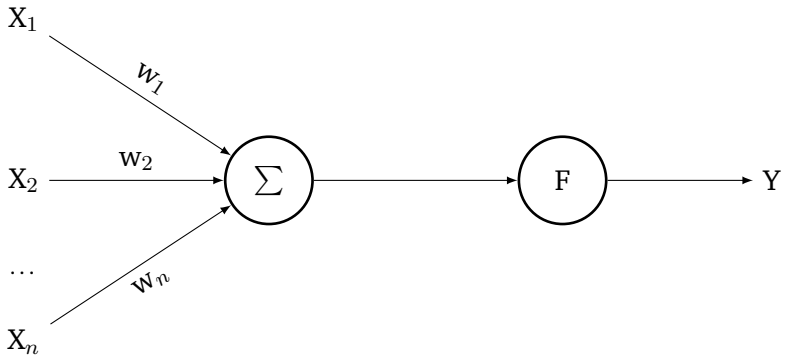


Figure 4.6: The McCulloch-Pitts model of a neuron, showing an activation function $F(x)$ acting on the sum of the weighted inputs to produce output Y

Up until this point all ANN architectures comprised a single layer of neurons. Minsky & Papert

(1969) demonstrated that single layer networks have an inherent inability to model non-linearly separable problems, such as the XOR gate. As a result of this discovery, ANN research stagnated in the 1970s; however it was rejuvenated in the 1980s by the advent of Multi-Layer Perceptrons (MLPs). These networks extend the existing perceptron by incorporating a *hidden* layer of neurons, allowing for more complex expressions to be represented, which can solve non-linearly separable problems; an example MLP is shown in Figure 4.7. With the addition of Hopfield networks, which implemented recurrency (Hopfield, 1982), and the backpropagation training algorithm (Rumelhart et al., 1988), the field of ANNs became a highly well researched area of literature once again. This momentum has continued to the modern day with research becoming more practical owing to increasing computational power alongside the use of distributed computing techniques, such as GPUs. Current trends include vertically stacking large numbers of hidden layers together to form so-called *deep* networks, which are currently the state-of-the-art in high-dimensional problems, such as image recognition (Krizhevsky et al., 2012; Simard et al., 2003; Lawrence et al., 1997).

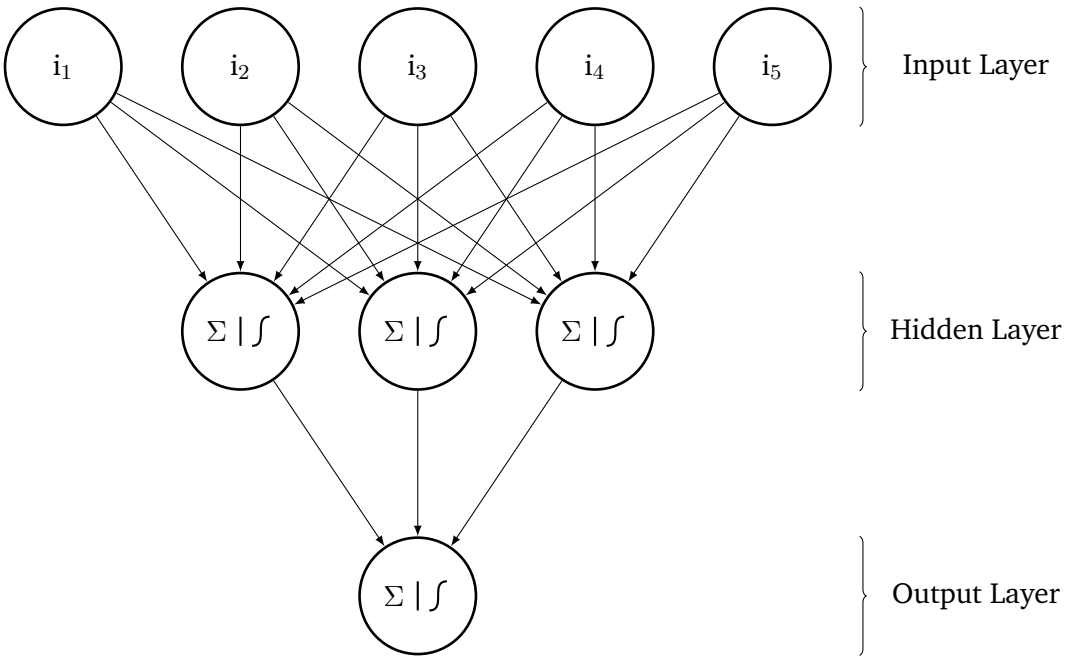


Figure 4.7: A Multi-Layer Perceptron (MLP), showing the 3 distinct layers of neurons

4.3.3 Multi-Layer Perceptrons

This section and the following ones provide further details on common ANN implementations, starting with the most basic feed-forward network—the Multi-Layer Perceptron (MLP). Figure 4.7 highlights the three structural components of a MLP: the input, hidden, and output

layers along with associated neurons. Determining the output of a network to a specified set of inputs constitutes making a *forwards pass* through the layers as follows: an input pattern—in the form of a feature vector for predictive modeling—is passed into the input neurons, which are connected via weighted edges to the hidden layer (weights not shown on diagram to avoid clutter). Each hidden neuron sums its weighted inputs and evaluates the output of this value after being input into an *activation function* (similarly to the McCulloch-Pitts model); functions which output sigmoid curves are typically chosen for this role, including the logistic function and $tanh$. The resultant *activation level* from each hidden neuron is subsequently passed into the output layer, whereby the output neurons calculate their own activation levels of their weighted connections in the same manner. The activation levels of the output neurons are the network outputs; only one output node is required for a binary classification problem, as a hard prediction can be made upon application of a threshold. For data sets with $c >= 2$ output classes, a network can be developed with c output neurons, each corresponding to a confidence score for that class that can be converted to a value with probability-like properties via the softmax transformation.

There are several ways to train a network to achieve the desired behaviour, where the desired behaviour is indicated by a *target* vector of expected values for each output node. For example, in a binary classification problem where the network has a single output neuron, the training signal would consist of a '1' or a '0' for each pattern depending on the actual class label. A range of behaviours can be produced by tweaking the model hyper-parameters, which are mostly associated with the network topology and include the number of hidden neurons and hidden layers. The primary objective of training MLPs, however, is to optimise the connection weights so that the network output best resembles the desired target signal. The gold standard algorithm for training MLPs is *backpropagation*, which, upon its introduction, led to a resurgence in ANN research. Backpropagation works by updating the connection weights from randomly initialised values to those which are discovered to have most closely approximated the desired output values, measured by minimising the total error. This is achieved by a two stage process, the *forwards pass*, followed by the *backwards pass*. The forwards stage simply comprises calculating the current output neuron activation levels using the manner described previously, resulting in a total error derived as the sum of the difference between the actual output and the target output for each output neuron. The backwards pass propagates the error back through the network much in the same manner as with the forward pass, just in the opposite direction. This allows for the calculation of the error signal for each computational neuron, which then have their weights updated via gradient descent

to minimise this value.

Unlike GP, MLPs have received some interest from the statistical modeling community owing to certain similarities they share with traditional linear models (Faraway, 2005). For example, the weights can be interpreted as model parameters to be optimised in the same way as the coefficients in a General Linear Model (GLM), likewise certain ANN practitioners incorporate a bias term into the network which performs the same role as the intercept of a regression model, furthermore the use of *weight decay* to limit large weights is comparable to ridge regression. However, a limiting factor in ANN take-up is the inability to guarantee achieving a global minimum with the backpropagation learning algorithm, unlike maximum likelihood estimators.

4.3.4 Recurrent Neural Networks

MLPs can be useful pattern recognition and classification models for traditional problems, whereby a model is required to learn to identify an output value from a supplied feature vector. However, as with most modeling techniques, they do not perform well when faced with high dimensional data, typically requiring some form of *feature selection* to build workable models. Furthermore, many traditional statistical methods are unable to adequately harness the additional dimension inherent in problems of a temporal nature. An extension of MLP to allow directed cycles between nodes—and thus providing short term memory—is termed Recurrent Neural Networks (RNNs) and solves this limitation, thereby allowing for models with greater inference capabilities than that found in traditional modeling techniques, in addition to providing benefits for other areas of machine learning such as control tasks.

The simplest form of a RNN is the Hopfield Network which consists of a single layer of fully connected neurons and typically uses an all-or-nothing thresholded activation function. More complex recurrent networks exist and are commonly used for problems requiring their particular pattern of connectivity, such as the Elman model (Elman, 1990) and the Jordan model (Jordan, 1997). These networks—shown in Figures 4.8 and 4.9—maintain a memory of previous neuron activation levels via a context layer; the inputs to the context neurons are taken from the hidden layer in the Elman model, and the outputs in a Jordan network. The outputs of the context neurons are made available to the hidden layer at the next time step, providing a means of saving network state. One commonality between these network architectures is that they can struggle to identify trends which are separated by a large number of time steps, Long Short-Term Memory (LSTM) networks are a form of RNN developed

by Hochreiter & Schmidhuber (1997) to solve this problem and have proven to be successful at this task. Extensions of backpropagation exist which allow for the training of RNNs, such as "backpropagation through time", however it can be challenging to interpret and implement such learning algorithms as they effectively un-roll the network at every timestep.

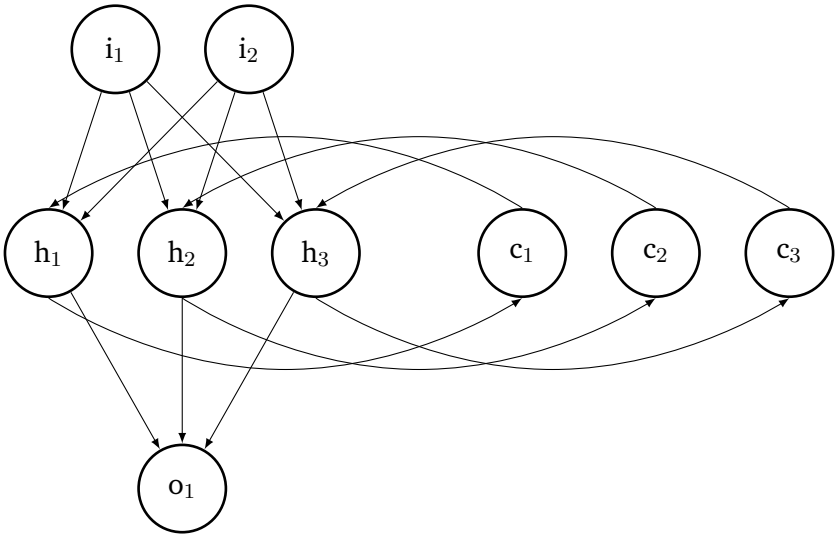


Figure 4.8: Elman networks include a context layer to maintain hidden layer state between time steps

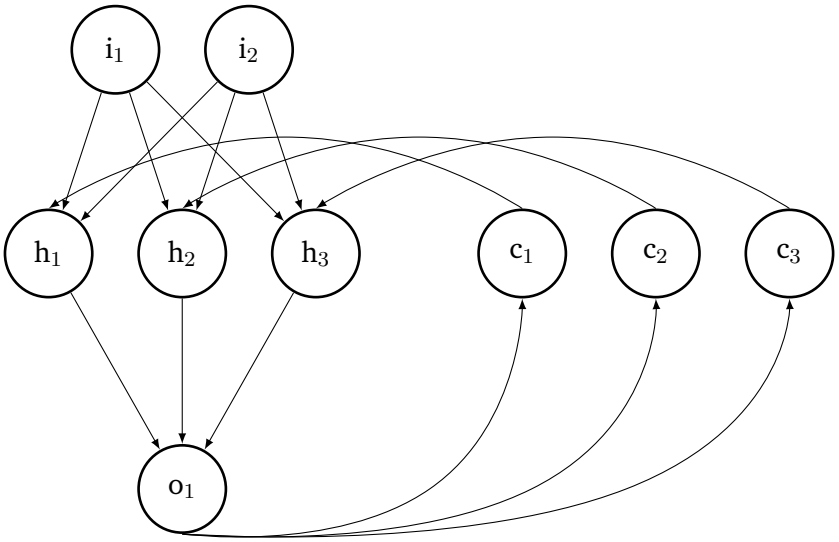


Figure 4.9: In Jordan networks the context neurons are connected to the output nodes

4.3.5 Reservoir Computing

The Elman and Jordan networks detailed above are well established models formed from a simple addition to the standard MLP, resulting in networks which have memory. In recent years, a new—and increasingly well researched—approach to RNNs has been developed,

termed *reservoir computing*. There are two primary differences with standard time-delay ANNs, the first being the replacement of relatively small discrete layers of neurons with a large *reservoir* of nodes. While there are sets of input and output nodes to map the data to the reservoir and provide a linear readout, the primary focus is on the reservoir dynamics. A large number of randomly connected neurons (with recurrent connections allowed) comprise the reservoir itself, typically implementing non-linear activation functions. Feedback from the output into the reservoir can be provided if the application warrants it, along with recurrent output connections.

The second main difference with simple recurrent networks such as the Elman or Jordan models, which just operate a simple time delay, is that only the weights of the output nodes are modified during training. This is achieved by implementing linear regression to model the training signal as a function of the reservoir activation states, and using the fitted coefficients as the output weight values. This enables a much simpler training procedure than those used for standard RNN techniques, such as backpropagation through time. Two common implementations of reservoir computing are Echo State Networks (ESNs) (Jaeger, 2001) and Liquid State Machines (LSMs) (Maass et al., 2002*b*). ESN typically use a sparsely connected reservoir of sigmoid nodes, while LSMs use a more biologically plausible neuron model by incorporating spiking neurons in the reservoir (Yamazaki & Tanaka, 2007). Figure 4.10 shows an example ESN.

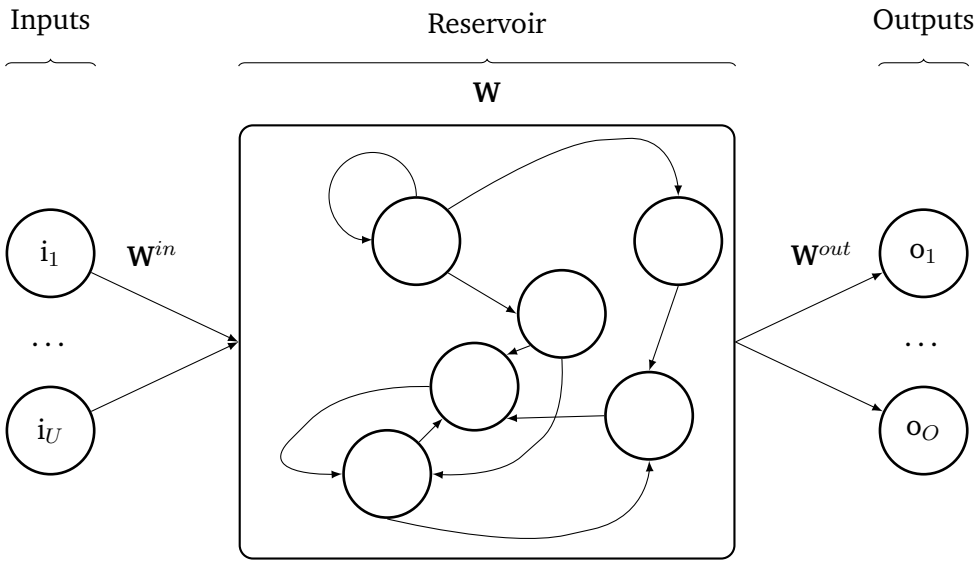


Figure 4.10: The reservoir of an ESN is sparsely connected with recurrent connections providing a means of maintaining state. Only the weight matrix $\mathbf{W}^{out}$ is optimised during training

4.3.6 Neuroevolution

While there are several possible learning algorithms for training ANNs depending on the specific choice of architecture, backpropagation is the most common choice for feed-forward networks, in addition to being used by RNNs with adaptations such as backpropagation-through-time. However, it still contains several limitations, including: the inability to guarantee it will converge on a global maximum; becoming trapped in local maxima; requiring a specified target signal and differentiable activation function; challenging interpretation of backpropagation-through-time; and for many network architectures the tuning of hyper-parameters that control the graph structure is still required. An alternative learning paradigm, which attempts to solve these issues, incorporates EAs as the primary training method in a field called *neuroevolution*. EAs have several advantages over backpropagation for training networks, as they are more able to escape local maxima, they can train a model to any specified fitness function rather than requiring a specific target signal, and via breeding operators they can modify the network in more ways than just the connection weights. However, these advantages come at the cost of additional computational time.

There have been numerous attempts at combining the global search capabilities of EAs and the distributed knowledge representation of ANNs, which have been reviewed by Yao (1993, 1999) and Floreano et al. (2008). In addition to optimising the connection weights, the use of EC techniques also facilitates the automatic optimisation of the network structure itself, and can even help to generate the most effective learning rule. EAs have also been deployed in tandem with backpropagation, with the EA identifying areas of interest in the global solution space, before using backpropagation to identify the local maxima. Two well known neuroevolution implementations are EPnet (Yao & Liu, 1997), which incorporates Evolutionary Programming (EP) to train feed-forward networks, and NeuroEvolution of Augmenting Topologies (NEAT) (Stanley & Miikkulainen, 2002), which grows a full network from just inputs and outputs using a GA. Both of these techniques optimise both the model parameters (in the form of connection weights), in addition to the topology. As EAs only necessitate a single measure of goodness to conduct a search, they are flexible enough to be applied to training RNNs with little modification; GNARL, developed by Angeline et al. (1994), is an example of such an approach.

4.4 CI Applied to Predictive Modeling

4.4.1 Overview

Many forms of GP and ANNs have been successfully implemented as learning algorithms to train classification models, in addition to traditional predictive modeling techniques such as those described in Section 3.2.3. CI algorithms typically form classifiers by a process of randomly generating initial models and subsequently improving them via an iterative search. The final models are typically highly non-linear and offer an alternative approach to standard statistical methods, although the search algorithms implemented by CI methods are often more computationally expensive.

The main advantage of applying CI to classification problems over using standard predictive modeling techniques is that the former approach can be much more flexible in the model representation, training approach, and also in terms of underlying assumptions concerning the data. Certain statistical models require assumptions about the data to be met, for example, logistic regression does not fare well with collinear data attributes and dependent variables, likewise the number of samples must be greater than the number of features present in linear discriminant analysis. Such algorithms often need large sample sizes for all assumptions to be met; CI algorithms have no such requirements, often only needing a guiding measure of model goodness. They are often more flexible in their model representation as well, being able to process data of varying formats rather than just discrete summary features. This has made CI techniques leading algorithms in many areas with complex multidimensional data, such as image recognition.

4.4.2 Managing Overfitting

As with models formed via traditional statistical learning algorithms, CI classifiers can be prone to overfitting to the training data and lacking in generalisation ability. This problem is exacerbated by the highly precise non-linear classification expressions formed as a result of a lengthy training procedure. The same techniques as used in predictive modeling to combat over-training—described in Section 3.3—can be implemented alongside approaches tailored to the specific learning algorithm.

Classifiers trained with iterative learning algorithms such as EAs and backpropagation can use the *early stopping* technique to reduce overfitting, comprising prematurely terminating the

training run before the iteration limit has been reached. The decision of when to stop the algorithm can be made based on the accuracy of the fittest model on a separate validation set—when this score starts to decrease the classifier is becoming overfitted to the training data and losing generalising capability. When analysing small data sets where subsetting a further validation set is not feasible, over training can be combated by simply using a lower iteration limit.

A concern with using CI learning algorithms is that they can produce an extremely thorough search, which, when combined with dynamically sized classifiers—such as expression trees—can result in highly specific discrimination rules with more terms than a standard linear model and large interaction depth. Similar to how decision trees are liable to overfit to the training set, so too are expression trees trained with EAs. While the size of GP evolved expressions has been shown to not have an effect on overfitting by Langdon & Buxton (2001*b*), Abbass (2001) demonstrated that increasing the number of hidden neurons in an ANN has a tendency to produce overly trained classifiers. He introduced the Memetic Pareto Artificial Neural Network (MPANN) algorithm to produce more generalised ANN classifiers by using an MOEA to simultaneously maximise the accuracy of the evolved classifier and minimise the number of hidden neurons.

4.4.3 Ensemble Building

CI techniques have been successfully applied to help build accurate ensemble classifiers in two major ways. First, ensembles have been formed using models trained by CI algorithms; ANN ensembles in particular have been the focus of a considerable amount of research (Hansen & Salamon, 1990; Krogh et al., 1995; Zhou et al., 2002). Initially such techniques involved training the base classifiers independently before combining with standard voting techniques. As CI models tend to be more complex; contain a greater number of parameters; and have a more involved training procedure than traditional predictive models, there are a large number of areas where diversity can be preserved when developing an ensemble. The second application of CI to ensemble classification is in optimising the ensemble selection and voting scheme from a collection of base models; a survey of such approaches from the literature now follows.

As previously discussed in Section 3.4.4, there has been some research in the literature into training the ensemble to further increase model accuracy, either by optimising the combination method or by fitting the base classifiers with the intention of combining these later in an

ensemble. This can be regarded as an optimisation problem with a large solution space, little insight into how best to achieve this, but with a simple quantifiable measure of success (ensemble accuracy). CI techniques are thereby well suited to training ensembles. One approach to ensemble generation which has proven popular is exploiting MOEAs to train a collection of classifiers which are both optimally diverse and accurate. A thorough survey of such techniques is described in Gu et al. (2015), although several of the most popular methods are detailed here.

GP Ensemble Voting

Langdon & Buxton (2001*a*, *b*) developed an ensemble training approach using GP to combine the votes of previously trained base models. In particular, a population of expression trees was evolved, whereby each individual is represented by five syntax trees constructed using a function set comprising pre-trained classification models. The output scores from each of the five trees were summed together to produce an ensemble prediction for each individual, effectively producing ensembles with five members. The output values took the form of continuous real numbers, whereby the sign represented the predicted class label and the magnitude the conviction, allowing a confident member to outweigh the other votes.

Under this framework, the EA is searching the solution space to find the optimum ensemble combination of the base models, rather than an individual classifier. Initially the function set consisted of standard C4.5 decision trees (Langdon & Buxton, 2001*b*), although ANNs and Naïve Bayes classifiers were also later attempted (Langdon & Buxton, 2001*a*), along with hybrid combinations of these different architectures (Langdon et al., 2002). Because each base model accepts the same inputs and returns an output value on the same scale, they can be used interchangeably in arbitrary GP expressions.

Furthermore, it was demonstrated that the performance of these ensembles was greater than Scott et al.'s (1998) *Maximum Realisable Receiver Operating Characteristics* of an ensemble classifier system (Langdon & Buxton, 2001*b*, *a*; Langdon et al., 2002). This approach is an example of training an ensemble by optimising the base classifier combination rather than using a standard linear voting method.

Negative Correlation Learning

Liu & Yao's (1999) Negative Correlation Learning (NCL) technique is a method which optimises

both the ensemble and its members at the same time. The authors' approach is to combine both the training of individuals and their combination to form an ensemble into one procedure. In doing so, the performance of the ensemble is directly fed back into the training process to ensure diverse individuals are trained. NCL extends the traditional backpropagation learning algorithm to include a penalty term for how similar the individual's errors are to the rest of the population, as shown in Equation 4.2, where $p_i(n)$ is the penalty term for an individual i on pattern n, $F_i(n)$ is the pattern output from i, and $F(n)$ is the ensemble output. It was originally tested on small ensembles of feed forward ANNs, but showed promising results (Liu & Yao, 1999).

$$p_i(n) = (F_i(n) - F(n)) \sum_{j \neq i} (F_j(n) - F(n)) \tag{4.2}$$

Evolutionary Ensembles with Negative Correlation Learning

Liu et al. (2000) extended NCL with an evolutionary search technique to produce their Evolutionary Ensembles with Negative Correlation Learning (EENCL) method, which combines both a global search (achieved with EP), and a local one (NCL is applied at each generation to optimise every individual). At each of these levels, pressure is placed on the individuals to diversify themselves, using the NCL extension to backpropagation during the local search, and a fitness sharing technique for the global training. For each pattern in the training set, the individuals that correctly predicted the class receive a fitness of $1/n$, where n is the number of individuals that correctly classified the sample. An individual's overall fitness is determined by summing the scores from each pattern, thereby encouraging individuals to classify patterns that few others have succeeded at. At the end of the evolutionary cycle, the population is divided into *species* using the k-means clustering algorithm with the fittest model from each species being promoted into an ensemble. EENCL thereby applies diversity preservation techniques at several stages, in both the global and local training of the population, followed by the ensemble selection criteria. However, it is computationally expensive to run such a hierarchical training scheme.

DIVACE

Chandra & Yao (2004) further developed the concept of optimising an ensemble's diversity with their Diverse and Accurate Ensembles (DIVACE) algorithm. DIVACE uses a similar

approach to EENCL, whereby an MOEA (based on Abbass's (2001, 2003) MPANN) drives a global search, and combines it with backpropagation applied at each generation to ensure every individual is operating in their local maximum. The first objective of the fitness function is to maximise accuracy, but the second objective aims to minimise the penalty function from NCL to promote diversity, rather than minimising the model size as in the original implementation. This produces a population of diverse individuals that are accurate, but have negatively correlated errors on the training set. DIVACE has been demonstrated to achieve better generalising ability than MPANN on a test set of benchmark classification problems, in addition to a smaller error rate on training data (Chandra & Yao, 2004, 2006*a*). The authors later extended this approach to incorporate a framework comprising a hierarchy of three levels of evolutionary training to form accurate and diverse hybrid ensembles (Chandra & Yao, 2006*b*), as well as developing a new similarity measure entitled *Pairwise Failure Crediting (PFC)*, which functions similarly to implicit fitness sharing by calculating the ratio between the number of incorrectly predicted patterns in common to the total number of incorrectly predicted patterns for a pair of classifiers (Chandra & Yao, 2006*a*). To obtain a global score of a classifier's similarity with respect to a population, this pairwise measure is averaged across each pair in the ensemble. Incorporating PFC as the second objective to be optimised alongside accuracy was demonstrated to result in more accurate ensembles than those formed with the original NCL formulation Chandra & Yao (2006*a*).

Others

McKay & Abbass (2001*b*) investigated three diversity preservation mechanics outside of the context of ensemble learning, in particular they compared implicit fitness sharing, NCL, and an adaptation of NCL termed root-quartic NCL, in a 6-multiplexer problem. They discovered that the new formulation of root-quartic NCL slightly outperformed implicit fitness sharing and regular NCL although at the cost of requiring an additional hyper-parameter to tune. This work was later adapted to the problem of evolving ANN ensembles (McKay & Abbass, 2001*a*) and improved over the original EENCL approach. NCL based ensembles have also seen use in a neural Learning Classifier System (LCS) based approach (Dam et al., 2008), considering three different voting schemes of majority voting, average, and winner-takes-all. They showed that incorporating ANNs into LCSs reduced the number of learned rules from the order of magnitude of thousands down to tens, in addition to demonstrating the benefits of the NCL diversity penalty function, provided adequate hyper-parameter tuning.

NCL was also the subject of research by Chen & Yao (2009), who incorporated a regularisation term to aid against over training after observing the tendency of NCL to overfit to noise in the data. This idea was later extended to produce an MOEA ensemble building technique which optimised three objectives: accuracy, the NCL penalty term, and the regularisation penalty (Chen & Yao, 2010). Using MOEAs to regularise ANN ensembles was also the focus of Jin et al. (2004), although rather than attempting to produce a diverse pool of classifiers for the optimal ensemble building environment, the ensemble selection process from a fixed pool was optimised by a weighting procedure. Oliveira et al. (2006) also investigated ensemble selection from an existing classifier pool using MOEAs.

Unlike the majority of these evolved ensemble techniques which explicitly optimised for diversity as an objective in an MOEA (see Gu et al. (2015) for a survey of these methods), Gagné et al. (2007) built ensembles from a standard single objective evolutionary run. To allow for diversity to be preserved amongst the population, they used a co-evolutionary approach whereby the fitness of each candidate solution is proportional to its success at classifying the most challenging patterns. This was achieved by weighting each pattern by the number of individuals which had correctly classified it, with an individual's fitness determined by the weighted sum of these values. This approach allows for diversity to be maintained with respect to having successes negatively correlated (unlike the PFC diversity measure which compares failure patterns). They also investigated the impact of building the ensemble from the final generation or holding an archive set incrementally added to at each generation, concluding that the former method produced the most accurate ensembles.

A more recent application of EAs to ensemble classification was developed by Bhowan et al. (2011, 2013, 2014), who applied a similar MOEA framework incorporating the NCL penalty to those used in DIVACE and MPANN to the problem of unbalanced binary classification. A common issue with modeling such data sets is that the modal class typically has more than twice the number of samples than the minority class, producing a situation whereby classifiers can achieve high levels of accuracy simply by predicting all observations as belonging to the model label and thereby not learning any meaningful discriminatory signals in the data. They investigate several areas of the ensemble building process, including the choice of MOEA algorithm between NSGAII and SPEA2, and the second diversity preserving objective, as either the original NCL calculation or PFC as developed by Chandra & Yao (2006*a*). The technique is successfully used to build ensembles that can accurately classify both the minority and the majority class, unlike more traditional machine learning methods such as Support

Vector Machines (SVMs) and Naïve Bayes. The results also indicate that the PFC diversity preservation objective presents slightly stronger ensembles than those formed using NCL, although there is no clear winner on each of the six data sets under investigation. The authors investigated the ensemble selection process; considering either including all the classifiers from the full Pareto front, discarding those who did not have a strong accuracy on the minority class, or performing stepwise member selection from the final generation at the end of training, with this latter approach proving most accurate.

Summary

In summary, there has been substantial research into the theme of exploiting the population based search of EAs to provide a group of trained classifiers from which to build an ensemble system. As evidenced from this literature review, the majority of these techniques have shared common characteristics, including: training ANN base classifiers, using the NCL similarity measure, typically employing MOEAs to explicitly optimise for diversity, and often using either average voting systems or a winner-takes-all approach. However, there has not been an overall survey which investigates all areas of ensemble building, including:

- The choice of base classifiers, including whether a hybrid ensemble can offer any advantages to a homogeneous one
- The means by which diversity is preserved throughout the training process—in these referenced studies it has been incorporated at the fitness evaluation level, either as a penalty term, or in the form of a secondary objective to be optimised in an MOEA setup
- The choice of similarity measure—most of the research in this field has employed NCL, or a variation of it such as PFC
- The ensemble selection process—how to choose the ensemble members from the large number of individuals generated throughout an evolutionary run
- The size of the ensemble
- The voting aggregation function—typically linear combinations have been used
- Whether to further train the ensemble combination stage

Table 4.1 summarises the surveyed evolved ensemble methodologies with regards to these aspects of ensemble formation. It highlights that the majority of these techniques have

Table 4.1: Comparison of previous evolved ensemble approaches

Study	Multiple Base Classifiers	Diversity Method	Similarity Measure	Ensemble Selection	Ensemble Size	Voting Strategy	Further Training
Liu & Yao (1999)	✗	Accuracy penalty	NCL	✗	✗	✓	✗
Langdon & Buxton (2001*b*)	✓	None	None	✗	✗	✗	✓
Liu et al. (2000)	✗	Accuracy penalty, fitness sharing	NCL	✗	✗	✓	✗
Chandra & Yao (2004)	✗	MOEA	NCL	✗	✗	✓	✗
Chandra & Yao (2006*b*)	✓	MOEA	NCL,PFC	✗	✗	✗	✗
Gagné et al. (2007)	✗	Co-evolution	Hardness weighted accuracy	✓	✓	✗	✗
Chen & Yao (2010)	✗	MOEA	NCL	✗	✗	✗	✗
Bhowan et al. (2013)	✗	MOEA	NCL,PFC	✓	✓	✓	✗
Bhowan et al. (2014)	✗	MOEA	PFC	✓	✓	✓	✓

relied upon explicitly optimising the diversity of the population by means of an MOEA with two primary objectives, both accuracy and diversity. Furthermore, in the majority of these algorithms, the diversity measure has been Liu & Yao's (1999) NCL, which measures the correlation of output scores assigned to each pattern from a neural network. Out of the non-MOEA approaches, diversity has been preserved in the population by means of manipulating the fitness function, penalising those individuals which score similarly to the rest of the population. Furthermore, the base classifiers used in the majority of the ensembles have been neural network based, which are commonly viewed as a black box. GP expression trees on the other hand can be more interpretable provided they are not grown too deep, but also can provide a means of inherent feature selection, as attributes that contribute to more accurate candidate solutions are more likely to be kept in the gene pool. This allows the user to obtain an indication of which of the data components are most discriminatory without having to employ a separate feature selection phase prior to modeling. In addition, hybrid ensembles—which have been shown to provide an additional means of maintaining diversity—have not been researched to a large extent by the EA community.

The work described in this thesis expands upon the application of EAs to ensemble building by considering traditional single objective varieties, and surveys the aforementioned aspects of ensemble formation in one body of work. Furthermore, the suitability of preserving diversity in

other areas of evolutionary learning than by penalising the fitness function will be investigated, in addition to hybrid ensembles.

4.4.4 Time Series Classification

As discussed in Section 3.5, feature design approaches to classification reduce the dimensionality of raw sequence data into smaller sized manageable feature vectors to be used for standard modeling. Several attempts at automating this process were discussed, although they resulted in a large number of features and did not produce interpretable systems. The primary consideration when developing an objective feature design algorithm is how to process the data without domain knowledge. CI techniques are often used in situations where there is a large search space and a lack of *a priori* knowledge concerning what an effective solution would look like. GP in particular is an attractive method for those wanting to develop a thorough automated process as it is flexible in its representation and behaviour, indeed, several feature design algorithms have been developed using GP in the literature. An alternative method is to model the underlying behaviours by means of a dynamical system, such as a RNN. This section summarises the research into time series classification using both of these approaches.

GP Strategies

Sharman et al. (1995) used GP to evolve adaptive signal processing algorithms, using arithmetic operators, time-delay functions, and a stack for retaining useful output values. With this method they were able to develop programs with similar functionality to RNNs and approximate digital filters, however, classification was not implemented. Parallel Architecture Discovery and Orchestration (PADO)—developed by Teller & Veloso (1995, 1997)—provided a framework for classifying input signals from a variety of forms including time series data, images, and video files. This was achieved by using domain specific function sets and a shared stack. Another algorithm, FIFTH (Holladay & Robbins, 2007; Holladay et al., 2007), was also designed to handle vector data of any form, and functions similarly to PADO with a shared data stack and signal processing function set. In contrast to these more general algorithms, Zeus (Eads et al., 2002, 2004) focuses on time series classification. Under Zeus, each chromosome represents a candidate feature vector, whereby each gene refers to a function selected from a set comprising both arithmetic operators and statistical means of aggregation. However, the authors showed that modeling the example time series data set using SVMs without any feature design produced significantly more accurate classifiers than with using

Zeus. Harvey & Todd (2014) developed a system called *Autofead* which leverages the power of GP to extract features from sequence data, via the use of a large function set comprising signal processing functions (including Fast Fourier Transform (FFT)) and statistical operations. An additional time series classification approach was developed by Xie et al. (2012), and was developed for the specific application of event (or anomaly) detection. The function set comprised windowed operations, with the window starting and ending indices specified by evolvable input parameters. Each windowed function also contained a parameter representing the operation to calculate on the segment from a choice of statistical functions such as mean, standard deviation, and skewness; the operation for each window was also optimised by the evolutionary algorithm. This approach was tested on simulated data in addition to being applied on a real world problem of anomaly detection in videos.

Lones, Smith, Alty, Lacy, Possin, Jamieson, Tyrrell et al. (2014) developed a GP based technique for classifying univariate movement data based on the shape of the waveform. Input data was segmented into arbitrary length windows, with the inputs to the expression trees comprising ratio offsets into the window and an arithmetic function set was employed to produce a summary feature from the window. The windows were evaluated independently, whereby the input nodes determined the value of the windowed waveform at the corresponding offset and used this as the value for that leaf node, before evaluating the entire tree to produce a scalar value for each window. To score each pattern, the mean window output to the expression tree was calculated. As detailed in Section 7.1, one of the approaches to classifying time series data in this thesis extends this method in several aspects.

In summary, there have been several previous approaches to time series classification using GP, mostly incorporating Digital Signal Processing (DSP) operators in the function set, allowing for scalar values to be obtained from the frequency domain. When operating in the time domain, stacks have been employed to facilitate evolved functions to navigate across the temporal dimension. An alternative strategy for time domain analysis has been to segment the data into windows, the method chosen by Xie et al. (2012) to select the window location and size through evolution is particularly interesting as it removes the possibility of incorporating bias into the window specification. For periodic data however, simple segmentation into the constituent cycles provides an objective natural means of windowing. A previous windowing approach based on classifying the shape of the waveform has shown promise and is extended in this work.

Dynamical Systems

The techniques for modeling time series data described in Section 3.5 involved summarising the raw data into discrete feature sets using aggregate functions such as the mean, standard deviation, or signal processing tools. The previous section discussed using GP to automate this process, with several of the studies incorporating stacks and time delay functions to navigate through the temporal dimension. While this approach can work and produce favourable results, it can be challenging to implement and interpret; however, there exist CI algorithms which provide a more natural means of modeling dynamic data which will be discussed now.

Algorithms such as the previously discussed RNN, Gene Regulatory Network (GRN) (Jakobi, 2003; Banzhaf, 2003), and Artificial Biochemical Network (ABN) (Lones et al., 2014), model biological processes in the body as networks, thereby allowing for recurrent connections providing short term memory alongside the possibility of using output feedback. Such techniques are termed computational dynamical systems (Stepney, 2012).

A standard predictive model—including the GP feature design approaches—is a mathematical function mapping inputs (a feature vector) to an output (either a label prediction or a series of scores or probabilities), whereby the output remains constant for a given set of inputs. Dynamical systems on the other hand provide stateful programs, adapting to changes in input data, and can be run online. As a result, they are frequently applied to control tasks and temporal forecasting (Quick et al., 2003; Lones et al., 2011, 2010; Williams & Zipser, 1989; Hunt et al., 1992; Nicolau et al., 2010; Kuan & Liu, 1995; Connor et al., 1994).

Such networks can be implemented for classification tasks too, although the majority of research concerning modeling time series data is concerned with forecasting rather than discriminating into discrete groups. Examples of classification applications in the dynamical systems literature have commonly used RNNs, such as Allen & Kamm (1991); Burrows & Niranjan (1994); Hüsken & Stagge (2003). A large consideration when using dynamical systems to classify sequence data is the manner in which to present the inputs into the system. Unlike the flexible representation of GP, recurrent networks are more fixed, with such approaches typically accepting one sample per timestep in the same manner as would be used for a forecasting application. Previous work by Lones et al. (2013) has highlighted the suitability for dynamical systems in the form of ABNs to model Parkinson's Disease (PD) from finger tapping movement data, however, the use of EAs to evolve complex biologically inspired networks requires considerable computational expense. Fitting a training set comprising multiple time series can thus be somewhat computationally expensive, which proves problematic when

performing model selection across a range of hyper-parameter values.

The emerging field of reservoir computing has demonstrated strong modeling capabilities when applied to problems with a temporal element, typically forecasting signals. For example, they have been successfully implemented to classify speech (Skowronski & Harris, 2006, 2007; Verstraeten et al., 2005; Maass et al., 2002*a*), predict stock market prices (Lin et al., 2009), predict the grammar in language tasks (Tong et al., 2007), and control robots (Salmen & Ploger, 2005). However, they have not largely been used either for medical analysis or on problems with movement data. Verplancke et al. (2010) incorporated ESNs into a study predicting the likelihood that a patient would require dialysis after being admitted to the Intensive Care Unit (ICU) given various biomarkers, including diuresis and creatinine levels. However, they discovered that while ESNs maintained reasonable levels of predictive power, they were outperformed by SVMs and Naïve Bayes (NB) classifiers. This application employed ESNs in a feature based time series classification approach, by reducing the dimensionality of the input sequence into a single scalar value, from which a classification can be made.

An alternative method developed by Chen et al. (2013) employed ESNs as kernel functions, by fitting a reservoir to each input time series and comparing the distance between the models themselves in a SVM framework to form the final classification. While this approach was shown to be competitive and more computationally efficient than the traditional Dynamic Time Warping (DTW) kernel function, it still necessitates the fitting of the linear readout weights for each time series in the data set, in addition to the overhead cost introduced by the distance calculations. In their work on classifying speech patterns, Skowronski & Harris (2006) use multiple reservoir readout filters, to target different parts of the input time series. The configuration of Verplancke et al. (2010) required the fitting of a single set of readout weights and so remains more efficient. In summary, dynamical systems can be applied to classification of time series either in the form of directly using the system output as the prediction in the form of a feature design method, or indirectly as the kernel function in distance based methods. ESNs offer great advantages for such applications owing to their rapid training time.

4.5 Conclusions

This chapter has detailed alternatives to standard predictive modeling learning algorithms, which take inspiration from natural biological phenomena to form objective searches to identify accurate models. As can be seen from this literature review, there have been multiple

attempts at implementing CI techniques for use in predictive modeling strategies, with this work focusing on their application to building ensembles and classifying time series data.

Ensemble classifiers are a popular area of current machine learning research owing to their ability to generalise well to unseen data by maintaining a set of classifiers which make their errors on distinct parts of the feature space, thereby allowing the voting stage to reduce the impact of individual incorrect predictions. As detailed in Section 4.4.3, EAs have been applied to ensemble building due to their inherent suitability as a population based search algorithm, with the majority of proposed algorithms being based on using MOEAs to explicitly promote both diversity and accuracy in the population. While this strategy is appropriate and has produced accurate models, it requires more computational expense than a traditional single objective EA and such techniques have not been picked up by the wider machine learning community, perhaps due to their unfamiliarity with these methods. There has been substantially less research activity into the use of single objective EAs for ensemble formation, which is unexpected as there is a significant body of research concerned with modifying EAs to maintain distinct niches in the population—one of the key goals behind ensemble formation. Furthermore, while many aspects of the ensemble building process have been individually investigated, including the use of hybrid ensembles, different diversity measures, the impact of voting strategies, and the concept of further training the ensemble, there has not been an overall review investigating all these effects simultaneously.

As identified in the previous Chapter (in Section 3.5), time series classification is an area of predictive modeling which typically employs domain knowledge to extract relevant summary features to be used in a standard learning algorithm, or uses computationally complex distance based classification methods such as DTW. Section 4.4.4 described alternative techniques for time series classification exploiting the flexibility of CI algorithms, in particular approaches using GP and dynamical systems have been surveyed. While this is a relatively new area of research, it has far-reaching consequences owing to the large amount of high dimensional multimedia data frequently being recorded.

The remainder of this thesis explores both of these issues in more detail, to establish whether EAs can be used to form accurate ensemble classifiers which can aid with PD diagnosis and whether CI techniques can provide an objective and effective means of feature design from time series data.

Chapter 5

Building Ensemble Classifiers using Evolutionary Algorithms

Contents

5.1 Introduction

Ensemble classifiers have become popular tools in machine learning research in recent years owing to their ability to produce more accurate and robust predictive models than an individual classifier. They primarily rely upon maintaining a set of accurate but diverse members; a large body of ensemble research in the literature concerns how best to achieve these criteria (as discussed in Section 3.4.2). Evolutionary Algorithms (EAs) are inherently well suited to forming classifier ensembles as they perform an explorative search of the solution space of potential predictive models—increasing the probability of finding models that fulfil both the accuracy and diversity criteria—in addition to maintaining a population of candidate solutions, providing the basis from which to build the ensemble. However, in the EA research community they are infrequently applied to this problem on their own, instead more complex algorithms are typically used, such as Multi-Objective Evolutionary Algorithms (MOEAs). Furthermore, EAs have not been largely adopted by the wider data mining community. This chapter investigates the suitability of single objective EAs for forming accurate ensemble classifiers with the addition of simple well established niching strategies. Three different areas of ensemble formation are considered, including the **diversity** management of the population of base classifiers, the manner by which the ensemble members are **selected** from the pool, and the method of **vote aggregation**.

5.2 Motivation

Referring back to the summary table of the surveyed evolved ensemble approaches in the literature (Table 4.1), it can be seen that the majority of these methods have used MOEAs to explicitly optimise for diversity. The specific diversity measure optimised by MOEAs is typically the Negative Correlation Learning (NCL) measure which calculates the correlation between an individual's (often a neural network) output for each pattern with the remainder

of the population. In cases where single objective EAs have been used, diversity has been preserved by manipulating the fitness function. In addition, while several of the main aspects of ensemble building have been considered individually, there are not any studies which have assessed multiple considerations when building ensembles, including:

- The impact of hybrid ensembles containing heterogeneous base classifiers
- The method by which diversity is preserved in the training process
- The similarity measure used by the diversity preserving mechanism
- The method of selecting the ensemble from the pool of candidate classifiers
- The size of the ensemble
- The choice of voting aggregation function
- The impact of further training the ensemble voting stage

This chapter thereby has three aims:

1. Determine whether traditional single objective EAs can produce accurate ensembles
2. Investigate the impact of preserving diversity in aspects of the evolutionary run that are not related to the fitness calculation
3. Study the above seven considerations of ensemble building in one framework to provide a thorough overview of which of these aspects provide most benefit

5.3 Experimental Setup

To investigate the above three aims, EAs were used to evolve populations of base classifiers from which ensembles were later assembled, considering multiple stages of the development process. The ensemble building process was summarised into three areas, each of which was considered in turn:

1. Efforts to preserve diversity during the initial classifier training
2. The method by which the ensemble members were selected from the EA population
3. The vote aggregation function

These areas are discussed in further detail in the following sections. Each combination of training parameters was assessed on each of the ten data sets detailed in Table 5.1, obtained from the UCI repository (Bache & Lichman, 2013). The data sets were selected as they all have dichotomous responses, tying them in with the overall goal of this thesis to detect the presence of Parkinson's Disease (PD). This benchmark suite consists of a varied range of problems with certain feature sets comprising entirely continuous values, while others include nominal attributes or a combination. In addition, a variety of data set sizes are represented; the largest (Chess) comprising 3196 samples, while the smallest (Hepatitis) is more typical of medical data sets, containing 155 observations. Each data set was subject to basic pre-processing, consisting of the centering and scaling of continuous attributes and forming $n - 1$ dummy binary variables for categorical predictors with n levels.

To assess the constructed ensemble classifiers' generalising ability, each combination of ensemble building parameters was resampled using ten repeats of ten-fold cross-validation, thereby reducing variance from both the resampling of the test set, and also from the stochastic nature of EAs themselves. To elaborate, each combination of data set, base classifier model, breeding algorithm, niching similarity measure, selection algorithm, ensemble size, and voting strategy, was evaluated on one hundred resamples, providing a full experimental setup analysing each possible interaction between parameters. An investigation revealed that data resampling is a greater source of variation in classifier accuracy than from the EA itself, therefore repeated k-fold cross-validation was determined sufficient to minimise such noise—for further details see Appendix A. The Area Under Curve (AUC) was selected as the evaluation criteria for the reasons presented in Section 3.2.2. Accuracy, in its strict definition of ratio of samples correctly predicted, was not employed as an evaluation criteria in the work described in this thesis, however, the term *accuracy* itself is used, in the context of referring to a classifier with strong discriminatory ability.

To test for statistical significance, both in this chapter and subsequent ones, non-parametric tests were used as the results have not been shown to be distributed normally. In particular, the Friedman repeated measures test is used to identify whether the choice of classification parameters has an impact upon the resulting model AUC values; if a difference is found then post-hoc analysis is carried out using the pairwise Nemenyi multiple comparisons test. When only two configurations are being compared, the Wilcoxon signed-rank test is used instead. This is the well recognised approach recommended by Demšar (2006).

Table 5.1: Data sets used in this study.

Data set	Instances	Total features	Continuous features	Categorical features
Breast cancer (Wisconsin)	699	9	9	0
Chess	3196	36	0	36
Credit (Australian)	690	14	6	8
Credit (German)	1000	20	7	13
Heart (Cleveland)	303	13	6	7
Hepatitis	155	19	6	13
Ionosphere	351	34	34	0
Liver	341	6	6	0
Pima Indians Diabetes	768	8	8	0
Sonar	208	60	60	0

5.4 Preserving Diversity during Model Training

5.4.1 Overview

One of the key considerations when building ensembles is how to develop a pool of classifiers that are individually accurate, but make their errors on contrasting training samples. While EAs cover a large area of the solution space during the search process, they typically converge on a local area resulting in a single best solution. For many applications this behaviour is desired; EAs are typically employed for optimisation problems whereby the aim is to find the optimal solution. A converging search is also ideal when using EAs to train individual classifiers, however, when the task is to generate a diverse population of models, a different approach is required. This section details aspects of the overall experiment concerned with the evolving of the base classifiers, including the use of diversity preservation techniques to form populations suitable for building ensembles from.

5.4.2 Base Classifiers

EAs are capable of training any classification model containing parameters through an iterative population based search. In this work, three different classifier representations were used to investigate whether there is a significant difference in their ability to classify data—with the null hypothesis suggesting there is not due to the *no free lunch* theorem. The use of multiple base classifier representations also allows for the evolution of heterogeneous populations; a mixture of model representations is one way in which diversity can be introduced into an ensemble classifier. The three base classifiers all form non-linear mappings of the input feature

vectors to a single continuous output score; in a clinical setting this can be interpreted as an additional bio-marker to be used as a diagnostic aid. A measure of discriminatory ability can be obtained from this value by employing Receiver Operating Characteristics (ROC) to produce the AUC score (as described in Section 3.2.2).

The three classification representations used in this work are:

- Evolved expression trees
- Cartesian Genetic Programming (CGP) graphs
- Multi-Layer Perceptrons (MLPs)

Standard Genetic Programming (GP) expression trees were implemented as they are the most common GP representation and have well established capabilities for predictive modeling. CGP graphs were included in the survey due to a desire to include a second GP method, and also as a result of its proven capabilities as a classification algorithm. The final base classifier model—MLP—was implemented to provide an alternative to standard arithmetic expressions. Furthermore, the field of Neuroevolution has demonstrated benefits over the standard back-propagation based learning algorithm.

Evolution Parameters

The EA used to train the base classifiers implemented a generation limit of one hundred and a population size of two hundred. The choice of a relatively low generation limit and large population size was made to encourage an explorative search of the solution space to increase the diversity of solutions found. The breeding phase consisted of selecting two parents by tournaments of size eight (with replacement) and then either applying either mutation or crossover (with equiprobability) to produce two offspring. These parameter values were determined by experimentation. The manner in which crossover and mutation were implemented was specific to the classifier representation, and is described in the following sections. Offspring were always entered into the next generation instead of competing with their parents, again to encourage explorative searches, while elitism of size one was also employed to ensure that the fittest individual from each generation survived and could pass on its genes even if it was not selected to be a parent.

The fitness function guiding evolution was the AUC of a classifier, selected for the reasons described in Section 3.2.2. To reduce bloat and encourage smaller solutions—which have been

shown to generalise better in addition to reducing computational time—parsimony pressure was applied above a threshold of fifty computational nodes, with a penalty term of 0.0001 AUC per additional computational node.

Evolved Expression Trees

Expression trees modeled the data sets using an arithmetic function set, allowing for non-linear mappings between the input feature vector and the single output score. The function set comprised the following operators $\{+, -, *, /, max, min\}$. Tree initialisation was achieved by a process of *ramped half-and-half* (as defined by Koza (1992)), alternating between producing fully grown trees at a maximum depth of three levels, and the *grow* method with a terminal probability of 0.3. Breeding was carried out in the manner as described previously, whereby two parents were iteratively selected using tournament selection. A coin flip decided whether to run crossover or mutation to breed two children at each step. The crossover operator employed in these experiments was a standard sub-tree crossover method. Point mutation was the other means of breeding, using a probability of 4% as determined from prior experimentation.

Cartesian Genetic Programming Graphs

As described in Section 4.2.4, CGP is a form of GP which uses a phenotype in the form of a directed graph as opposed to the traditional expression tree representation. The CGP models used in these experiments utilised the same arithmetic function set as the expression trees; while some implementations of CGP allow for nodes with variable arity, the architecture used in this work limited function arity at two. Crossover operators are not used in CGP, as removing nodes from their original position in the graph would lose all associated context and thus not provide any significant benefit. Instead, the resulting expressions are tweaked solely by mutation, which in this implementation was in the form of a point mutation operator allowing both the choice of functions and network connections to be modified.

Multi-Layer Perceptrons

The field of Neuroevolution (introduced in Section 4.3.6) optimises Artificial Neural Networks (ANNs) using EAs rather than the traditional backpropagration algorithm, allowing both the connection weights and topology to be modified. In the setup used for these experiments,

single hidden layer MLPs were initialised with a number of hidden neurons dependent on the number of attributes (a) and classes (C) in the data set (shown in Equation 5.1), resulting in an initial pyramidal structure for all problems. The activation function used by hidden and output nodes was the sigmoidal error function.

$$N_{hidden} = \frac{a + C}{2} \tag{5.1}$$

One of the strengths of ANNs as a computational tool is the manner in which the decision making process is distributed amongst the numerous individually weak neurons. This is in contrast to tree-based GP, which combines higher level function nodes processing different aspects of the data into a single output. Thus, while combining sub-trees from two parents can produce effective offspring, such a procedure performed on two MLP parents can remove the established distributed knowledge, producing a weaker offspring (Yao, 1999). As with the CGP base classifiers then, only mutation was used to tweak the network expressions. Two mutation operators—*topology_mutation(x)* and *weight_mutation(x)*—were implemented, with a coin flip deciding which to use for each parent. *topology_mutation(x)* added an additional neuron to the hidden layer with a 40% probability and removed one in all other occurrences, thereby increasing pressure on producing smaller more generalised networks instead of large overfit solutions. The weight mutator, *weight_mutation(x)*, acted as a simple point mutation operator, modifying the connection weights with the same overall 4% probability as defined in Section 5.4.2.

Hybrid populations

The final population type consisted of evolving a mixture of expression trees, CGP graphs, and MLPs simultaneously with the aim of training a heterogeneous population from which a hybrid ensemble could be readily developed. The population was initialised with equal numbers of each model type but with no further preservation techniques implemented, leaving the potential for one or more of the representations to be removed from the gene pool. Since the standard population size of two hundred is not a multiple of three, the EA population size was increased to two hundred and one for the hybrid experiments, allowing for an even initial distribution of base classifier architectures, with sixty-seven of each.

Inter-species crossover is meaningless without the context of the original expression, and so only mutation was implemented for the training of heterogeneous populations. Rather than

selecting two parents at each selection phase, just one parent was picked from the population using tournament selection, with the model-specific mutation operator being used to produce an offspring. The same representation specific mutation operators and probabilities were used as in the homogeneous case.

5.4.3 Niching Operators

An alternative means of forming a diverse population is to manipulate the selection, reproduction, replacement, and fitness calculation phases of an EA to ensure that a variety of behaviours are present in the population. This is the aim of niching strategies (see Section 4.2.5 for further details), which preserve individuals operating in different areas of the solution space to form a multi-modal search, as opposed to a traditional EA seeking a single global optimum. Four different niching algorithms were implemented alongside a standard EA (as described in Section 5.4.2) to investigate whether these simple lightweight techniques offered any benefits for ensemble building. As discussed in Section 4.4.3, there have been previous approaches for ensemble building which have incorporated diversity preserving strategies by manipulating the fitness function to penalise similar individuals. The approach taken by this work is instead to consider niching strategies which work at the breeding stage of an EA—comprising the selection, reproduction, and replacement phases—in an effort to promote diversity more implicitly, while still rewarding highly accurate individuals. As a result, the fitness sharing niching strategy was not considered. Singh & Deb (2006) provides a comparison of a variety of niching strategies for their use in multi-model optimisation. Since their focus is on determining strategies for use in problems with a discrete known number of local optima, rather than ensemble generation whereby the aim is to evolve a diverse population as possible, the results are not directly transferable to this approach. However, they highlight the ability of deterministic crowding and restricted tournament selection to generate well separated niches in the population and will thereby be included in this investigation into their suitability to ensemble generation. Probabilistic crowding is also included as a more exploitative adaptation of deterministic crowding, and Species Conserving Evolutionary Algorithm (SCEA) is selected for the opposite reasons, since it tends to form a more exploitative search. In their survey, Singh & Deb (2006) also indicate their proposed modified clearing strategy has success in this area, however, it comes with an increased computational time and so was not considered for inclusion in this study.

Summaries of the selected niching algorithms, including their parameter values (as selected by

Table 5.2: Niching methods used in these experiments along with their hyper-parameter values

Name	Abbreviation	Hyper-parameters	
		Name	Value
Deterministic Crowding	DC	None	NA
Probabilistic Crowding	PC	None	NA
Restricted Tournament Selection	RTS	w	8
Species Conserving Evolutionary Algorithm	SCEA	σ_s	0.6
No niching	EA	tournament size	8

preliminary investigations), are laid out in Table 5.2. The other evolutionary settings were the same as with the standard EA, for example, Deterministic Crowding (DC) utilised the same generation limit and population size, the only difference being the method in which selection and replacement were realised.

5.4.4 Similarity Measure

All the niching methods assessed in this work require a quantifiable measure of similarity between two candidate solutions. The original implementation of crowding defined similarity in terms of *structure*, determined by the Euclidean distance between the two competing genotypes. However, as the intention of using niching methods for this application is to result in a population of classifiers which make their errors on different data instances, *functional* similarity measures might be more appropriate. In this work, both structural and functional similarity measures are employed by the niching algorithms to determine whether rewarding diversity in the desired outcome (in their behaviour) is more effective than preserving structural diversity in the hope this leads to uncorrelated outputs.

Correlation of Outputs

The first similarity measure assessed how well matched the output scores from each classifier were for each data pattern, similarly to NCL. However, unlike NCL which penalised based on Pearson's correlation coefficient, the implementation in this study incorporated the Spearman non-parametric correlation coefficient instead since it is unlikely for the scores from two non-linear models to be linearly related. This method explicitly attempts to fulfil the ensemble criteria of incorporating individuals that make their errors on different samples.

Features in Common

When training base classifiers for later use in an ensemble, the objective is to produce a set of models which make their errors on distinct parts of the solution space. Using the output correlation method described above, classifiers are developed which explicitly meet this criterion. An alternative approach, however, is to promote diversity in other ways to meet the optimum ensemble environment implicitly. As evidenced from the literature review, a common approach for ensemble building trains the base classifiers over subsets of the data set's feature space, thus building models which identify distinct patterns in the data. Since all three of the base classification representations contain an element of implicit feature selection, a potentially useful similarity measure is to determine how many data attributes are common to multiple models.

Both GP expression trees and CGP graphs maintain a discrete set of data features used in the expression, whereby attributes proven to be discriminatory are included in the model, enabling a similarity measure calculated by the number of features in common. MLPs, however, do not have a binary process of including a feature, instead, all the possible attributes are known to the network which can select the degree to which each input is used by manipulating its connection weights to the hidden layer. Thus, the feature similarity measure for MLPs calculates the average weight assigned to each input attribute by the hidden neurons, and uses the Euclidean distance to obtain a measure of likeness with a second network.

5.4.5 Summary

Three different areas concerned with the evolving of base classifiers were investigated, the choice of model architecture, the use of niching techniques, and the means by which similarity is assessed by the niching methods, all summarised in Table 5.3. The experiment will investigate whether either of the two diversity preservation techniques—using heterogeneous populations and niching strategies—aid ensemble accuracy, in addition to determining the impact of the choice of similarity measure on the effectiveness of the niching methods.

5.5 Constituent Member Selection

5.5.1 Overview

Over the course of an evolutionary run, a large number of candidate classification models

Table 5.3: Diversity preservation techniques investigated in this study

Aspect investigated	Options
Base classifier	GP expression tree
	CGP graph
	MLP
	Hybrid population
Niching method	Standard EA
	DC
	PC
	RTS
	SCEA
Niching similarity measure	Output correlation
	Features in common

are fitted to the training data with varying degrees of success, producing a large pool from which an ensemble can be assembled. An important consideration when building ensembles is the manner in which the members are selected from the available group, in addition to the number of models picked. This section details a variety of approaches for these aspects of the member selection process, and how they were implemented in the overall experiment.

5.5.2 Forming the Classifier Pool

The first consideration concerns how to form the pool of ensemble members from the candidate solutions produced throughout an evolutionary run. As the EA ideally should result in a collection of individually accurate classifiers, selecting the ensemble members from the final generation may seem appropriate as the search will have converged on a high performing region of the solution space. However, by discarding every generation of solutions except the final one runs the risk of losing potentially useful models. An alternative approach is to archive a set number n of individuals at each generation and append these to the classifier pool at the end of the run. Extending this approach by increasing n equal to the population size would result in an extremely large pool of potential ensemble members, increasing the chance of finding well separated models albeit at considerable computational expense and requiring a large amount of storage space.

Gagné et al. (2007) considered this issue in their work on adapting EAs for building ensembles, which used a fitness function based on co-evolution to promote diversity. The fitness function rewarded individuals which could successfully classify the 'hardest' patterns in the data set

rather than overall accuracy. They concluded that selecting the ensemble members solely from the final generation of the EA resulted in more accurate ensembles than from selecting the members incrementally throughout evolution. As a result, in this work the ensemble will be built solely from the population at the end of the EA run.

5.5.3 Ensemble Size

Preliminary testing, as well as other research (Maclin & Opitz, 2011), has indicated that ensembles with around twenty diverse members appear to be the most accurate. The classifier pool comprised two hundred individuals (as this is the population size—see Section 5.4.2), allowing for a large range of ensemble sizes. In this experiment ensemble sizes were taken from the set $\{5, 10, 20, 50, 100, 200\}$.

5.5.4 Selection Criteria

Once a pool of classifiers has been developed and the number of models to pick has been identified, the final step in building an ensemble is to use a selection criteria to identify the ensemble members. Ideally, the use of diversity preserving techniques such as niching or heterogeneous populations will have produced a well balanced final generation of classifiers so that the actual selection process does not impact too much on the ensemble accuracy. However, this is not always the case. In this work four different selection criteria have been identified and tested. Three approaches based on picking the most diverse individuals in the pool are included, along with an accuracy measure. An attempt at selecting the ensemble members as the most accurate individuals in k-means clusters was also investigated in preliminary work, but did not offer any advantages for its increase in computational time. The following paragraphs detail the four selection methods investigated.

Elitism

Elitism simply selects the n most accurate candidate solutions from the final generation of the EA to be used in the ensemble. As only accuracy–and not diversity—is considered when using elitism, it requires that the base classifiers are already well separated and make their errors on distinct parts of the training set, else there is little advantage to be gained from using an ensemble of highly accurate but well correlated classifiers.

Output Correlation

The remaining three ensemble selection criteria take the opposite approach and select the n most globally diverse classifiers from the population without regard to their accuracy, where global diversity is measured as the average pairwise similarity with the other classifiers in the pool. The first similarity measure to be investigated is using the Spearman rank correlation coefficient of the classifier's outputs to the training set, similarly to how it was used in the niching methods (as discussed in Section 5.4.4).

Features in Common

The other similarity measure used by the niching algorithms measured likeness between classifiers as a function of shared data attributes. For GP expression trees and CGP graphs this was calculated by the number of features used in common by both individuals, while for MLPs it was determined by the Euclidean distance between the two average input weight vectors. This pairwise similarity measure was also implemented as part of the ensemble selection process, whereby the n most dissimilar individuals in terms of their use of data features, were promoted to the ensemble. For hybrid ensembles this selection criteria was not used as it is not possible to quantify a suitable value of the similarity between an MLP and an expression tree in terms of data attribute usage, for example.

Number of Functional Nodes

The final similarity measure used as an ensemble selection criteria was the number of functional nodes in the model. The amount of computational terms in a predictive model has an impact on the model's complexity and its likelihood of overfitting to the training set. By maintaining a range of the number of terms in the discrimination functions, it was hoped that the members would form their predictions in different ways. For both expression trees and CGP graphs this measure was equal to the number of function nodes in the expression, while for MLPs it was calculated as the number of hidden neurons.

5.5.5 Summary

Table 5.4 details the two aspects of ensemble selection which were investigated, alongside the various options for these parameters. The aim of these experiments was to determine whether,

given a large collection of classifiers, the method in which in the ensemble is formed has an impact upon its accuracy. The trade-off between diversity and accuracy was also examined, with three of the selection strategies solely focusing on picking dissimilar members, alongside one based on accuracy. Since diversity preserving methods were utilised during the base classifier training, it could be the case that incorporating similar strategies in the selection phase does not offer any further advantages.

Table 5.4: Investigated aspects of ensemble selection

Aspect investigated	Options
Ensemble size	5
	10
	20
	50
	100
	200
Selection criteria	Elitism
	Output correlation
	Features in common
	Number of functional nodes

5.6 Voting Scheme

5.6.1 Overview

As detailed in Section 5.4.2, three different base classifier representations are being investigated in this set of simulations. Each of these evolved models acts as a non-linear function approximator outputting a single continuous score—rather than a discrete class label prediction—opening access to a large variety of linear and non-linear voting aggregators (discussed in Section 3.4.3). A range of ensemble vote combination techniques were assessed to investigate two objectives:

1. Determine whether *training* ensembles offers any advantages
2. Investigate if non-linear vote aggregators are more effective than more simple linear methods

This section details the six different ensemble voting techniques which were assessed, comprising a combination of linear and non-linear methods, and trained and un-trained.

5.6.2 Linear Aggregators

Two linear voting methods were implemented, an averaging system and a weighted average. These are analogous to the popular Majority Vote (MV) and Weighted Majority Vote (WMV) algorithms used by models that predict discrete labels. Under the WMV voting scheme, each classifier's weight (w) is dependent upon its fitness rank (r), and the number of ensemble members (L), as shown by Equation 5.2.

$$w_i = \frac{L - r_i + 1}{L} \tag{5.2}$$

5.6.3 Training Ensembles

Training an ensemble (see Section 3.4.4 for further details) is a process which takes an existing ensemble and further optimises its make-up by means of a training algorithm to improve classification ability and robustness. Genetic Algorithms (GAs) have been successfully applied to the problem of feature selection, by evolving bit-string chromosomes indicating which features to include in the model (Yang & Honavar, 1998; Opitz, 1999; Siedlecki & Sklansky, 1989). This approach has been adapted to select which classifiers to include in an ensemble's voting stage from a given pool (Ruta & Gabrys, 2005; Dos Santos et al., 2009; Sirlantzis et al., 2002; Gabrys & Ruta, 2006), whereby the chromosome represents which members' votes will be included in a majority vote system. These experiments investigate this technique in greater detail by analysing the effectiveness of training the ensemble in conjunction with the base model training and selection process. The following two approaches are taken to produce two evolved linear voting systems.

1. Evolving a bit-string GA to select the members votes to include in an average vote

2. Using a weighted average voting system whereby the weights are optimised by a floating point GA

These two methods provide trained counterparts to the standard linear average and weighted average voting systems, and will be used to asses the impact of training the ensemble.

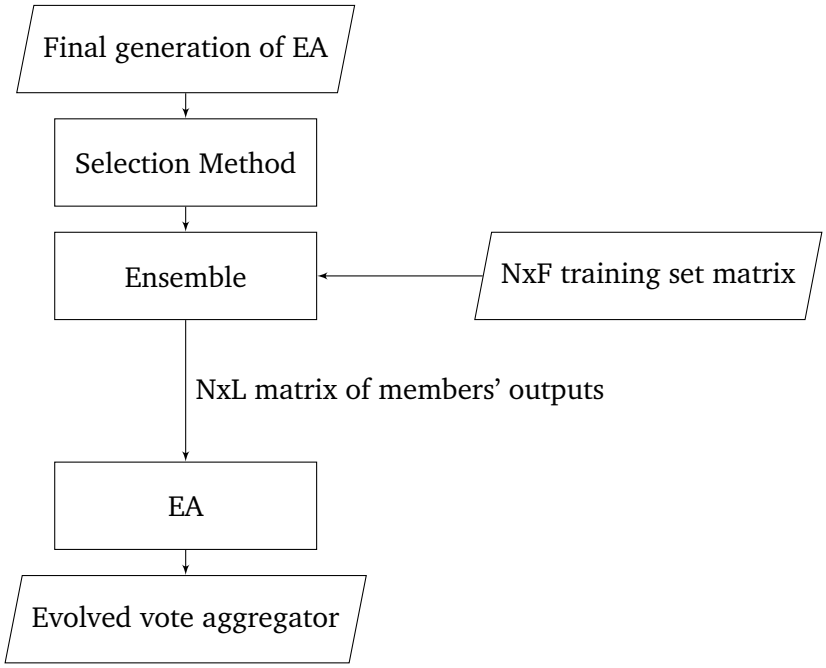


Figure 5.1: Training Evolved Vote Aggregators involves building a model of the ensemble outputs on the original training samples

5.6.4 Non-linear Voting Functions

When using ROC to evaluate classifiers outputting a continuous score, there exists a large number of possibilities of how to combine votes to produce the ensemble output—as discussed in Section 3.4.3. One approach investigated by Langdon & Buxton (2001*a*, *b*)—see Section 4.4.3 for full details—is to treat the base members as functions operating on the data patterns, before combining the outputs of these functions in a non-linear manner. This approach is extended in these simulations by forming a new data set comprising the ensemble members' outputs on the original training set. This process, shown in Figure 5.1, reduces the dimensionality of a data set with N patterns from F, to L, where L indicates the number of base classifiers in the ensemble.

Once the secondary feature extracted data set has been developed, any classification algorithm can be used to model the ensemble's votes. For the same reasons as being chosen as base classifiers, GP expression trees and MLPs have been selected to perform this task. Both of these representations provide a non-linear combination of ensemble votes, and will be trained through evolution, albeit in a slightly different manner to the training of the base models. Since the goal of this approach is to build an optimal ensemble, rather than an explorative multi-modal search to build a diverse population, a more elitist learning algorithm is required. An evolutionary strategy was used for this optimisation task, with a (1 + 4) configuration and

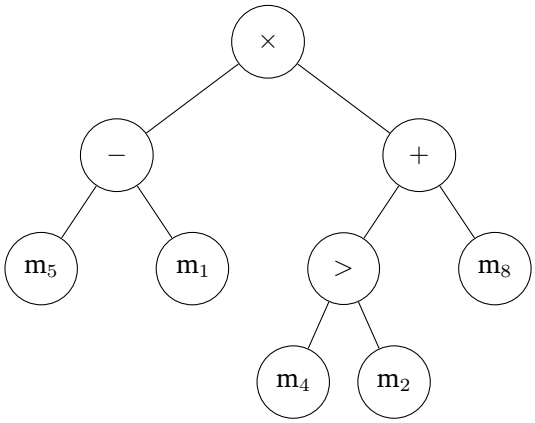


Figure 5.2: Expression tree EVAs use an arithmetic function set to combine members' scores

a 20% mutation rate being run for 4000 generations. Previous investigation indicated these parameters worked well for an elitist search which is still able to break out of local optima. These vote combiners are termed Evolved Vote Aggregators (EVAs) throughout the rest of this chapter for convenience, the following two paragraphs provide further detail on their implementation.

Expression Tree Evolved Vote Aggregator

As with the expression trees used as base classifiers, the expression tree EVAs are derived from an arithmetic function set—shown in Figure 5.2. Conditional functions were also preliminarily assessed but resulted in less accurate predictions than using simple arithmetic operations. The same breeding operators were employed as for the initial population breeding, namely sub-tree crossover and point mutation.

Multi-Layer Perceptron Evolved Vote Aggregator

The MLPs used to model the secondary feature extracted data set comprise the same architecture as that of the initial base classifiers, whereby a single layer is initialised with the number of hidden nodes indicated in Equation 5.1 and the sigmoidal error function is used by every neuron, as shown in Figure 5.3. Likewise, there are two mutation operators which modify both the network topology and connection weights, one of which is picked for each network with equiprobability. See Section 5.4.2 for further details.

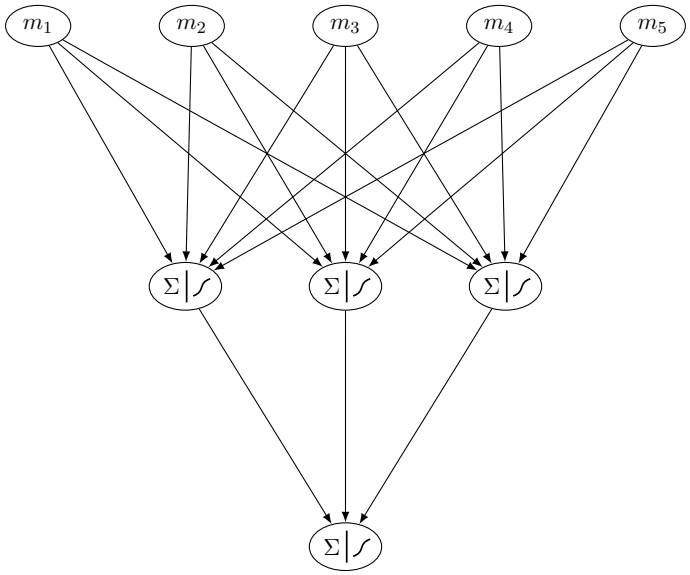


Figure 5.3: Extracted data features are passed into a MLP EVA during a secondary classification phase

Table 5.5: Investigated aspects of ensemble voting

Aspect investigated	Voting scheme
Standard non-linear voting methods	Average Weighted-average
Evolved linear voting methods	Evolved average Evolved weighted-average
Evolved non-linear voting methods	Expression tree EVA MLP EVA

5.6.5 Summary

Three different groups of ensemble voting strategies are being investigated with two implementations in each, totalling six vote combination strategies as detailed in Table 5.5. These experiments will investigate: whether training simple linear ensembles is beneficial, the impact of using feature extraction to form non-linear aggregators, and whether such non-linear techniques offer any advantages over simple linear ones.

5.7 Results

5.7.1 Overview

Experiments were run to investigate multiple facets of building ensemble classifiers using EAs, divided into the following three areas:

- Training the classifier pool
- Ensemble member selection
- Ensemble voting system

This section discusses the results of each of these three areas in turn, note that each possible combination of ensemble building parameters was evaluated to provide maximum insight into any possible interactions between ensemble building configurations. The evaluation criteria used throughout this chapter for comparison is the AUC of the developed ensemble on the validation folds, providing an estimate of the model's generalising ability. Note that in the following results there are observed cross-validation AUCs less than 0.5, despite Section 3.2.2 detailing that such scores are still in themselves evidence of better than random guessing and can be improved by inverting the classifier output. These odd results are explained by the model achieving an AUC of greater than 0.5 on the training set, but less than this baseline on the validation set (or vice-versa), indicative of a poorly calibrated model. This situation could have arisen purely owing to chance as a consequence of the large number of simulations, or due to a large difference in training and validation set samples from a small sized challenging data set. Every combination of ensemble building characteristics was evaluated on each of the ten data sets using the previously described ten repeats of ten-fold cross-validation, resulting in a total of 2834000 validation fold scores. This large sample size explains the wide range of AUC results, as due to chance some models will not make a good fit, in addition to certain combination of ensemble properties being ineffective.

5.7.2 Results Visualisation

In this section, and all future experimental chapters in this thesis, the results will be visualised in the same manner allowing for an overview of the distribution of the scores for each element under investigation, as well as enabling a rapid comparison to determine if any difference between two competing setups has statistical significance. In particular, boxplots are used to

Table 5.6: Base classifiers used in the study

Model architecture	Abbreviation
Expression tree	*gp*
Multi-Layer Perceptron	*ann*
CGP graph	*cgp*
Mixture of all three	*hybrid*

display the spread of AUC values from cross-validation folds (training scores are not displayed as by themselves they do not provide any indication of a classifier's practical performance). Since boxplots can be configured in a variety of manners, this section will briefly clarify the specific configuration employed in this thesis. The 'hinges' at the lower and upper bounds of the box are placed at the first and third quartiles respectively, while the whiskers extend to $\pm 1.5 \times IQR$ of the box; points beyond this range are plotted individually as outlier points. Furthermore, 'notches' are displayed at $\frac{\pm 1.58 \times IQR}{\sqrt{n}}$, facilitating an approximate 95% interval to compare whether the difference between two medians is statistically significant or not. Observe that this notch is present in all the following boxplots, although due to the large value of n in the main results (as each combination of ensemble building parameters is evaluated) this notch is very close to the median. The use of notches to enable a fast comparison between two medians is described by McGill et al. (1978).

5.7.3 Building the Classifier Pool

Three aspects of the evolving of the base classifiers were investigated:

1. The choice of classifier representation
2. The use of niching breeding techniques
3. The choice of similarity measure used by the niching methods

Tables 5.6 and 5.7 summarise the various options for the first two of these aspects under investigation, along with abbreviations used throughout this chapter. Each combination of these parameters was tested across all the selection and voting techniques, producing a thorough overview of the utility of each possible breeding parameter.

Table 5.7: Summary of niching techniques

Niching algorithm	Abbreviation
Deterministic Crowding	*dc*
Probabilistic Crowding	*pc*
Restricted Tournament Selection	*rts*
Species Conserving Evolutionary Algorithm	*scea*
Standard EA	*standard*

Base Classifier Model

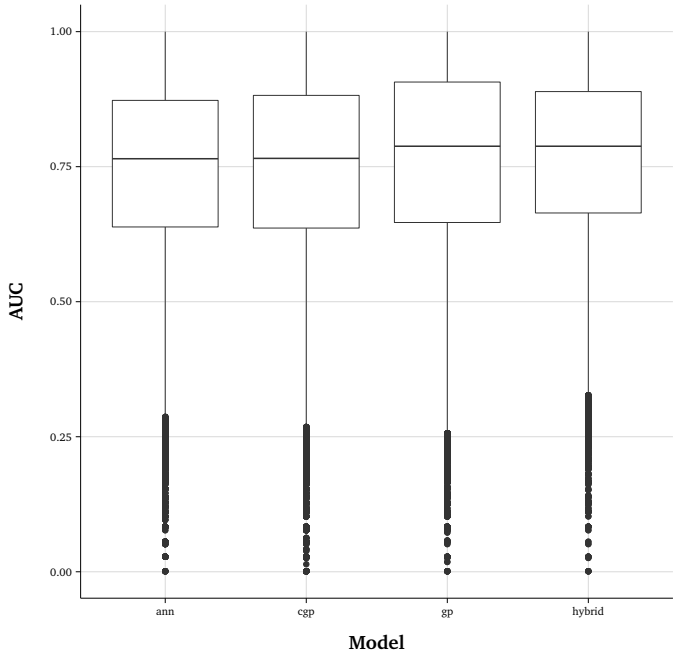


Figure 5.4: Base classifier comparison

Figure 5.4 displays the AUC of ensembles built from four different base classifier populations across all the validation folds, the large number of simulations accounts for the observable vast range of AUC values. It highlights that overall, ensembles built from GP expression trees are the most accurate, with there being little difference between the other two homogeneous populations comprising MLP and CGP models. The differences between each classifier representation was determined to be statistically significant at the 0.1% level from post-hoc testing.

Hybrid populations do not offer any significant advantages over using a homogeneous collection of base members, despite their additional source of diversity. A possible explanation is that the three architecture types were not evenly represented in the population, analysis

revealed that on average the dominant classifier type represented 67% of all individuals, double the 33% proportion that would be found in an evenly distributed split, indicating that the dominant model representation had a tendency to force out weaker ones from the population.

However, there were relatively few occurrences of the weaker two architectures becoming extinct and resulting in a homogeneous population by the end of the evolutionary process—this scenario occurred in only 1.4% of simulations. A form of conservation may be useful to help maintain a balanced heterogeneous population throughout the evolutionary process, which could aid ensemble accuracy.

Niching Method

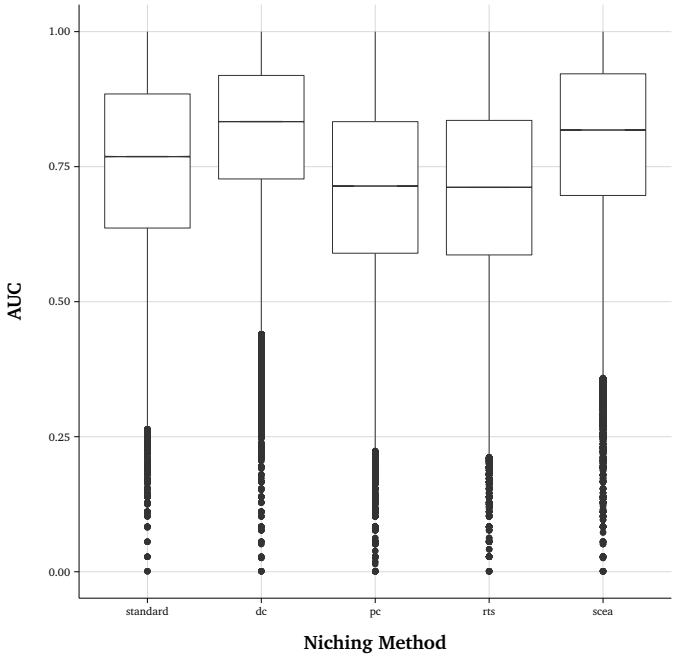


Figure 5.5: Niching algorithm comparison

The ensemble AUC scores for each niching algorithm are displayed in Figure 5.5. Overall, the Deterministic Crowding (DC) and SCEA niching techniques produce the most accurate ensemble classifiers, more so than those built from standard EA breeding operators. However, using a niching breeding operator is not always the most effective ensemble development choice, as both Probabilistic Crowding (PC) and Restricted Tournament Selection (RTS) result in ensembles which are on average less accurate than using standard tournament selection. Hypothesis testing indicated that the difference in the ranked AUC distributions for each

pairwise comparison of breeding operator was statistically significant at the 0.1% level.

Alongside the AUC, an important measure of ensemble efficacy is how much they improve over using the strongest member classifier; selecting the most individually accurate member from a pool could be considered a naïve voting strategy. A measure of *improvement* was derived, calculated as the percentage difference between the ensemble AUC and that of its most accurate base model, where a greater value indicates an ensemble is more effective than the fittest member. Table 5.8 details the mean improvement across all the simulations for each of the five breeding methods. This measure is particularly useful when combined with the ensemble AUC scores to determine which breeding algorithms are effective at producing single accurate classifiers, and which are more suited to ensemble building.

Table 5.8: Average AUC improvement of the ensemble over its fittest member

Breeding algorithm	Mean improvement (%)	Standard deviation
standard	-3.57	16.57
dc	-0.78	12.25
pc	-2.47	67.09
rts	-2.60	18.93
scea	-2.96	14.43

The most striking result is that all the breeding algorithms have a negative mean improvement, indicating that on average simply using a single classifier trained by the EA would have been more effective. The explanation for this unexpected finding is discussed later on in Section 5.7.5. Aside from this, the table indicates that, perhaps surprisingly, DC is the most efficient algorithm for developing ensembles. Since PC functions in much the same way as DC, albeit replacing the direct tournament for survival with a probabilistic one, it could be expected that PC would result in a population of well dispersed, but potentially weaker individuals than DC, thereby having a greater chance of building an ensemble more accurate than its base members. However, this was not the case, due to either the ensemble accuracy being lower than expected or the base models having a higher AUC than would be expected from a population evolved using such an explorative search.

The RTS method has a very similar improvement value to PC, along with highly comparable ensemble accuracies, however, these two algorithms have considerably lower AUC values than using a standard non-niching EA despite having nearly double the improvement over the fittest member. This indicates that these algorithms result in a very explorative search of the solution space to produce a diverse population, at the expense of accuracy. The standard EA appears to have the opposite problem, it produces strong accurate base classifiers but the ensembles offer

little advantage. Despite being a niching algorithm aiming to enable a multi-modal search of the solution space, SCEA performs very similarly to the more elitist search of the standard EA. It appears to produce highly diverse base members, despite the ensemble not having as large an impact as seen in the other niching algorithms. Importantly, all the niching algorithms offer greater improvement over their base members than the standard EA, highlighting the effectiveness of this approach for ensemble building. Overall, DC seems to offer the best compromise between diversity and accuracy amongst the base classifiers.

Similarity Measure

The four niching breeding methods implemented two different means of quantifying pairwise *similarity*: one based on the correlation between rank output scores, and the other comprising a measure of the features in common used in the competing models. The effectiveness of each of these two similarity measures across all niching algorithms is shown in Figure 5.6. It highlights that while there is not a large difference between the two approaches in real terms (albeit one where $p < 0.001$), comparing models on the data attributes they have incorporated is slightly more effective than how they rank the data set. This is an interesting finding, as the output correlation approach is explicitly attempting to build an ensemble which meets the criteria of having its members make their mistakes on distinct parts of the training set and so could be expected to be most effective.

When combined with the niching versus standard EA results, these findings suggest that while implementing selection and reproduction strategies to enforce diverse individuals is on the whole beneficial, the precise manner by which this is calculated does not have a large impact upon ensemble accuracy.

5.7.4 Ensemble Selection

Two factors of ensemble selection were under investigation for their effect on the overall accuracy:

1. The selection criteria
2. The number of constituent members

The ensemble sizes being investigated are values in the set $\{5, 10, 20, 50, 100, 200\}$, while the selection criteria are summarised in Table 5.9. These aspects were assessed on all the base

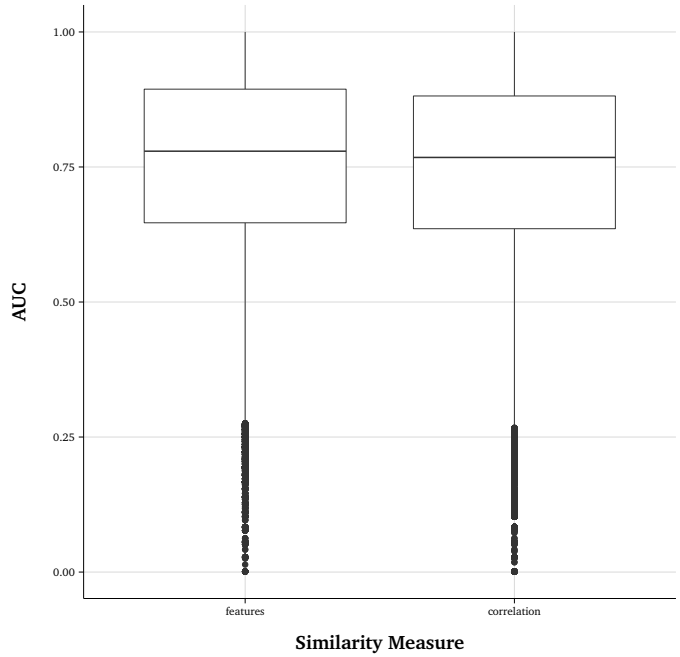


Figure 5.6: Comparison of niching similarity measures

Table 5.9: Ensemble selection strategies investigated in these simulations

Technique	Abbreviation
Elitism	*elitism*
Data attributes used in the model	*features*
Correlation between scores	*correlation*
Number of computational nodes	*functional*

classifier populations and across every voting strategy.

Selection Strategy

The four different ensemble selection measures are compared in Figure 5.7, indicating that the elitist approach of selecting purely the most accurate classifiers from the pool is more effective than a novelty search of picking the most diverse individuals. Out of the similarity measures, *correlation* is the most accurate on average, illustrating that explicitly selecting members due to their behaviour characteristics is more effective than choosing models on a structural basis. This could be due to the hypothesis put forward by Hansen & Salamon (1990) that the most effective ensembles are those whose members make their errors on differing parts of the data set, which the *correlation* method attempts to quantify.

The difference in ensemble accuracy between the similarity measures is not as large as between

them and the *elitism* technique, indicating that as with the niching results (Section 5.7.3), the choice of quantifiable diversity measure is not largely significant in absolute terms, however statistical hypothesis testing indicated that the choice of selection function is significant at the 0.1% level.

The same measure of ensemble *improvement* over the fittest member as was used to analyse the choice of niching strategy has also been calculated for the selection methods, with the results shown in Table 5.10. These scores reinforce the superiority of the *elitism* selection method, although again highlight that on average across all the combinations of ensemble building parameters, the fittest member is more accurate than the combined ensemble prediction.

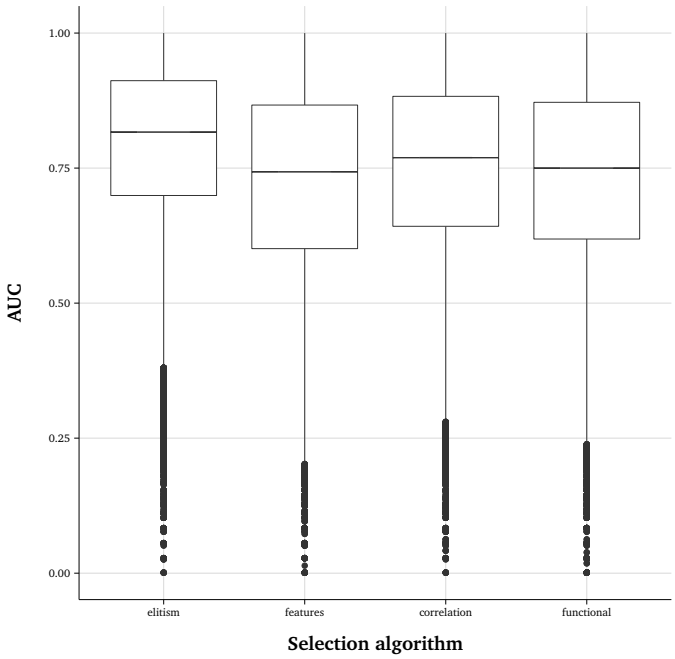


Figure 5.7: Ensemble selection strategy AUC scores

Table 5.10: Average AUC improvement of the ensemble over its fittest member across the selection strategies

Selection Method	Mean improvement (%)	Standard deviation
elitism	-1.51	14.81
features	-2.45	17.09
correlation	-2.10	60.33
functional	-3.60	17.38

Ensemble Size

Figure 5.8 shows an interesting trend with regards to the impact of ensemble size upon accuracy as the optimal number of base classifiers appears to lie within the range 50–100, larger than previous research indicates is optimal. However, it is important to place these results in context. All the ensemble selection strategies except for *elitism* are not guaranteed to include the single most accurate model in the pool, thereby increasing the need for a larger ensemble to increase the chances of selecting it. Figure 5.9 shows the relationship between ensemble accuracy and size for ensembles built using the *elitism* approach only. Under these circumstances, twenty members provides the optimal accuracy, in line with previous findings. In these simulations every pairwise comparison of ensemble size had significantly different AUC distributions, at minimum at the 5% level except for the comparison between 20 and 50 members.

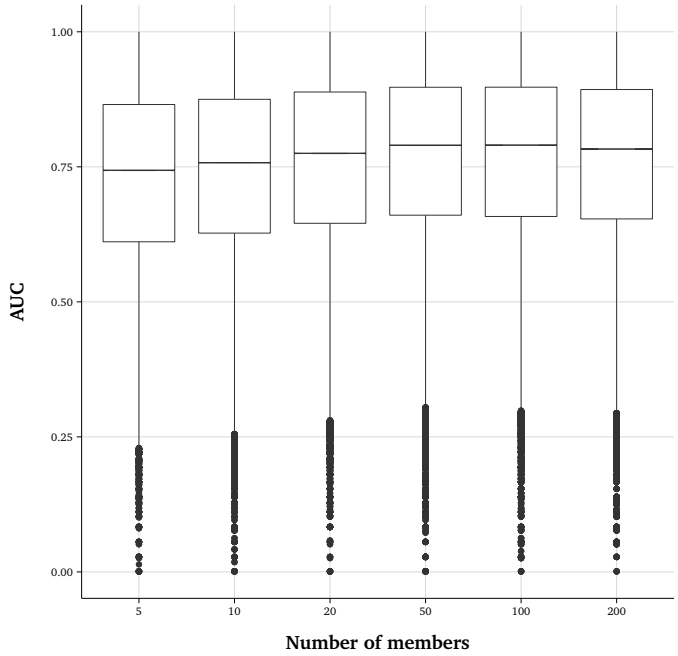


Figure 5.8: Ensemble size AUC across a range of sizes

5.7.5 Voting Aggregators

There were six different vote combination methods implemented, listed in Table 5.11 along with their shorthand names for easier referencing. Each of these approaches was tested on every combination of initial population and ensemble selection strategy, so that each voting strategy is assessed on the same ensemble members allowing for a thorough and fair

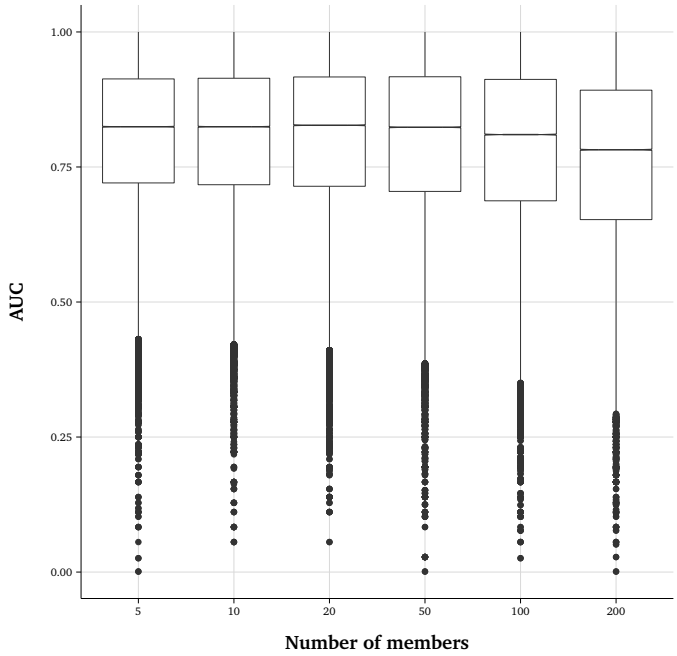


Figure 5.9: The impact of ensemble size when using the *elitist* selection strategy

Table 5.11: Investigated aspects of ensemble voting

Technique	Abbreviation
Average	*avg*
Weighted-average	*wavg*
Evolved average	*ev.avg*
Evolved weighted-average	*ev.wavg*
Evolved expression tree	*gp.eva*
Evolved MLP	*ann.eva*

comparison between them.

Linear Combiners

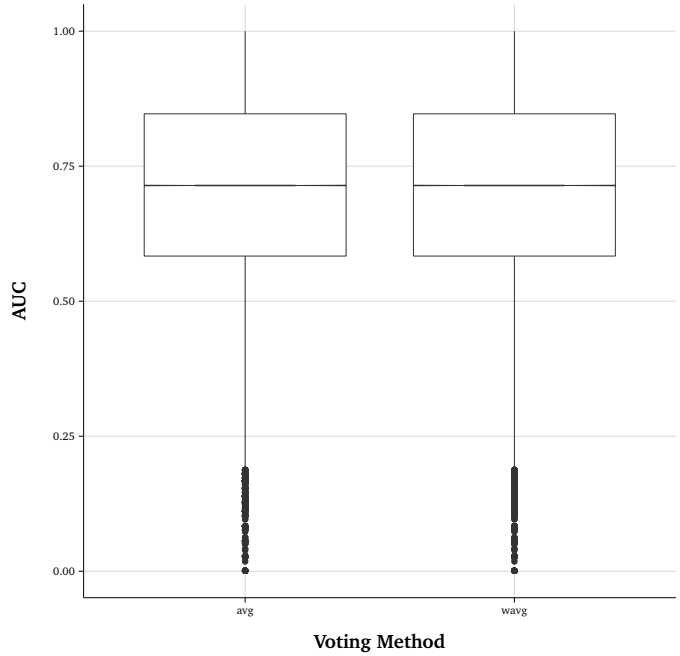


Figure 5.10: Comparison of AUC scores from the two linear voting methods

Figure 5.10 compares the performance of the two standard linear voting methods: the average and the weighted average. It is interesting to note that ensembles score extremely similarly using either of these approaches (p=0.717), this could indicate a limitation of the rank proportionate technique used to derive the weights for such ensembles—it remains to be seen whether using a fitness proportionate method would yield better results.

Training Linear Combiners

Table 5.12: Average AUC improvement of the ensemble over its fittest member across the linear voting strategies

Voting Method	Mean improvement (%)	Standard deviation
avg	-8.01	46.31
ev.avg	3.74	12.39
wavg	-8.01	46.31
ev.wavg	2.81	12.74

The comparison between the standard linear voting methods and their evolved counterparts (displayed in Figure 5.11) demonstrates the considerable benefit of training the ensembles using GAs; for each of the voting methods, training the ensemble produced a statistically

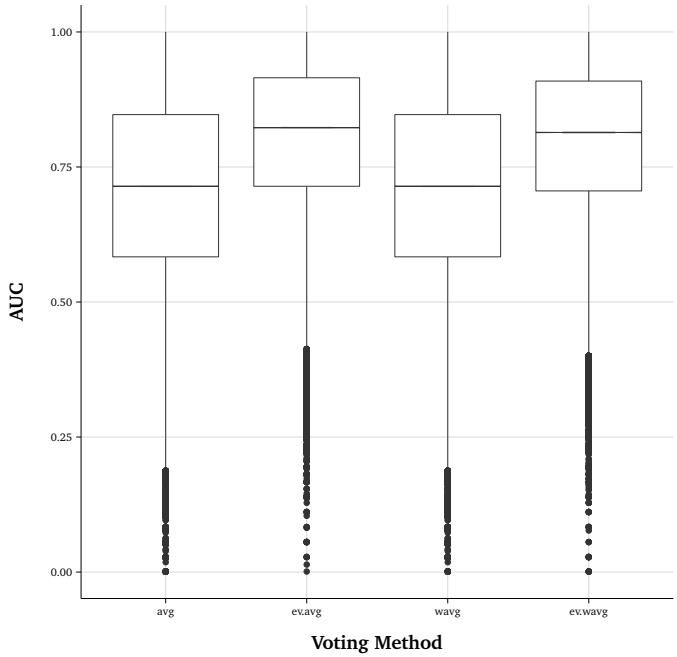


Figure 5.11: Results from all the linear voting schemes

significant improvement at the 0.1% level. In addition, it can be observed that selecting the members to include in the averaging process is slightly more effective than evolving weights themselves. This is an interesting finding as it might be expected that the greater finesse permitted by optimising continuous weights would allow for a more precise ensemble than one which is trained to either use or discard members' votes.

Previously, it has been highlighted that overall ensembles score less highly than their fittest members. The reason for this unexpected finding is displayed in Table 5.12, which shows the average *improvement* scores for each of the voting methods. While the *avg* and *wavg* ensembles have negative improvement scores, the trained versions do offer a significant increase in AUC. This provides practical information for building an evolved ensemble, in that the most significant gains in AUC result from using a trained ensemble, using the final population without any further training is likely to hinder model accuracy.

Linear versus Non-linear Aggregators

Having observed that using a GA to train the linear voting ensembles produced a significant improvement in ensemble accuracy, it would be interesting to see how these methods compare to training the ensemble with a highly non-linear Evolved Vote Aggregator (EVA). The two EVA methods (GP expression tree and MLP) were assessed on the populations formed using the

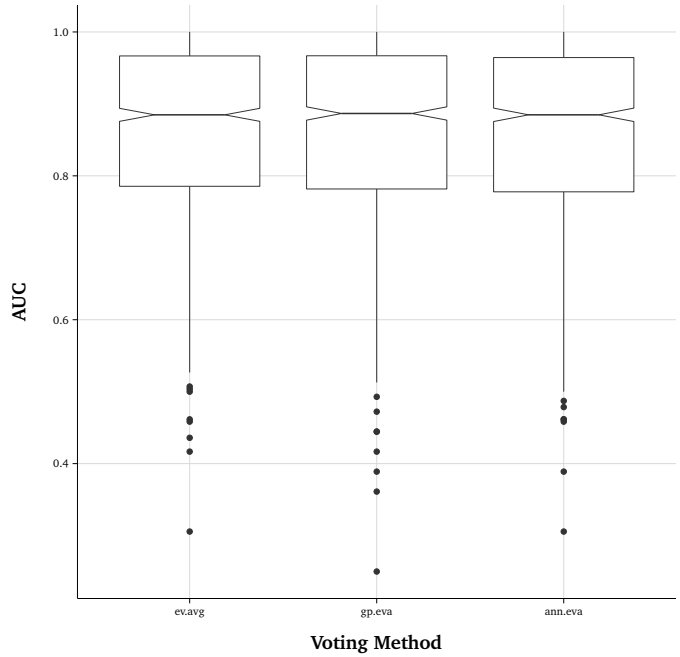


Figure 5.12: Results from the comparison between the best linear voting method and non-linear Evolved Vote Aggregators

most effective ensemble building characteristics discovered earlier in the section, comprising expression tree base classifiers, DC breeding, and were selected using the *elitism* strategy with a size of twenty. This enables a more thorough comparison between the voting schemes using the ensemble building strategies that would be selected for a real world problem.

Figure 5.12 displays the results of the most accurate trained linear ensemble (*ev.avg*), against the two non-linear EVAs. It reveals that using more complex non-linear vote combiners offers no significant advantage (hypothesis testing showed no statistically significant differences) over a simple bit-string GA. Between the two EVA types, there is little difference.

Table 5.13: Overtraining scores of the voting methods

Voting algorithm	Mean percent overtrained
avg	7.42
wavg	7.42
ann.eva	7.36
gp.eva	7.16
ev.wavg	6.84
ev.avg	6.34

Since the trained ensembles are being optimised directly on the same training set that the base members were fitted to, it would be useful to investigate to what extent they are being

overtrained. In addition, this could be a potential cause of the relatively poor performance of non-linear EVAs, as non-linear expressions have a larger solution space to search and as a result can fit strongly to training data. A measure of overfitting was derived, calculating the percentage difference between an ensemble's training set score and its AUC on the validation fold. The mean overfitting scores for each voting method are displayed in Table 5.13, where a positive value indicates that the model achieved a higher AUC on the training set than the validation data.

Unsurprisingly, all the voting methods have higher training set AUCs than achieved on the test sets, it would be extremely unusual if a model could consistently score better on an unseen data set than the one it is fitted to. All the voting methods overtrained to similar amounts, the EVA techniques to a slightly greater degree than the trained linear methods but not by a large margin. Interestingly, the *avg* and *wavg* voting techniques overfitted to the same proportion as the EVAs technique, despite not containing any additional training after the base classifiers had been developed. Therefore, these results indicate that training the ensemble does not introduce a significant amount of overfitting to the model.

When building a classifier for a specific application—particularly one to be used by medical clinicians as a diagnostic aid—it is important to consider how interpretable the model is. Using linear vote combiners improve over non-linear EVAs in two key areas: not only are they more accurate, but their inner workings are more decipherable owing to being a simple averaging of the constituent members. Adding a second non-linear modeling stage to the model (after the initial base classifiers have been evolved) would be challenging to justify to a clinician when a simple average achieves similar results for the aggregation. Furthermore, complex models are more susceptible to overfitting, and so it is for these reasons the linear combiners are preferred.

5.8 Ensemble Generalising Ability

One of the greatest benefits that ensemble learning provides over using a single classifier is the ability to produce a robust model which can generalise well to unseen data. Individual classifiers, particularly on small data sets, are often in danger of overfitting to the training samples. Since this thesis discusses modeling real world medical data sets—where recruiting just fifty patients for a study can prove time consuming and logistically challenging, yet alone five hundred—this benefit of ensemble learning is particularly attractive. This section

investigates the generalisation ability of the evolved ensembles, with a focus on small data sets to establish their suitability for use in modeling medical diseases.

5.8.1 Analysis

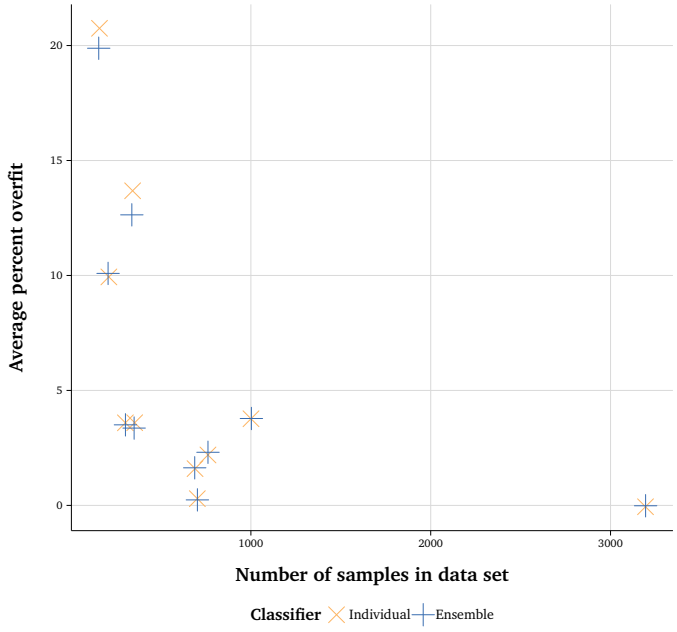


Figure 5.13: Overfitting behaviour of classifiers

The optimum parameters for building ensembles from EAs have been discovered to comprise GP arithmetic expression trees, a Deterministic Crowding (DC) breeding strategy using the number of features in common as a similarity measure, from which twenty members are selected by an elitist approach to form the ensemble. The most effective voting system for an applied model was the evolved linear average approach, given its short training time, its comparable accuracy with non-linear techniques, and its interpretability. Since this ensemble technique involves further training the model, it is likely that it too will present some degree of overfitting—indeed this behaviour has previously been observed for the evolved ensemble methods (see Table 5.13).

To illustrate this, Figure 5.13 plots the previously described measure of overfitting against the number of samples for each of the data sets used in this study (as detailed in Table 5.1), for both individual classifiers and ensembles. On some data sets the amount of overfitting exhibited by both ensembles and individual classifiers is so similar that the points are overlapping, however, overall, the plot highlights that smaller data sets—particularly those with less than

five hundred samples—are more susceptible to overfitting to the training patterns for both ensembles and single classifiers. It appears that there is an exponential relationship between the number of patterns and the amount of overfitting, indicating that the primary focus when building any predictive model should be to collect as much data as possible.

However, the true test of an ensemble is not how much it improves on its training set, but how advantageous it is over using a single classifier. Previously, a measure of *improvement* has been used for this purpose, comprising the difference between the ensemble and the most accurate base member's validation fold scores. The average improvement on each data set for this optimal set of ensemble building parameters has been plotted in Figure 5.14.

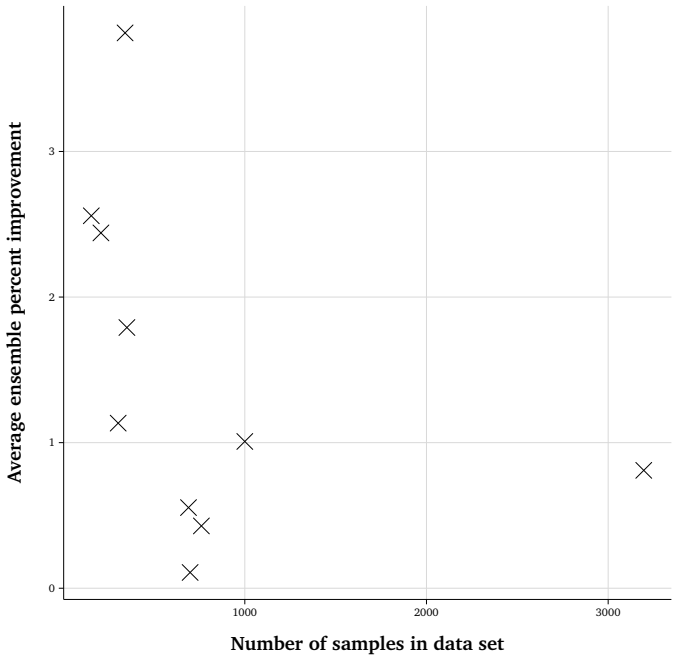


Figure 5.14: Ensemble improvement as a function of data set size

It highlights a negative exponential trend between data set size and ensemble improvement, indicating that ensembles provide the greatest benefit for data sets at the greatest risk of over training. This ability of evolved ensembles to generalise better than single classifiers is particularly attractive when modeling medical data, and so this ensemble building approach will be used in future work for analysing PD movement data. While the relationship between number of training samples and ensemble improvement looks relatively well established, it is drawn from a small sample size of ten data sets; it would be useful to increase to include a greater number of data set sizes, particularly in the range of 1000–2000 samples, to establish the nature of the relationship between ensemble size and improvement offered. In addition,

despite evolved vote aggregators offering the greatest improvement over a single model, they are still susceptible to overfitting during the training process. It would be beneficial to investigate means of reducing this phenomenon to further increase ensemble accuracy.

5.9 Comparison with Other Learning Algorithms

5.9.1 Evolved Ensemble Techniques

In Section 4.4.3, a number of previous approaches to building ensemble classifiers using evolutionary approaches were detailed and discussed. The major difference between these techniques and that introduced in this work is the manner in which diversity is preserved in the population; in the works cited in the literature review, diversity controls were typically implemented at the fitness evaluation stage, explicitly rewarding diverse individuals by penalising those well correlated to the rest of the population either as an additional objective in an MOEA or by a penalty applied to the cost function. In this work, however, efforts to preserve a diverse and explorative search have been focused on the selection and replacement phases of an EA to implicitly build a diverse population of accurate individuals. An additional difference is that most of the evolutionary approaches detailed in Section 4.4.3 have combined global evolutionary learning with backpropagation at the individual level to perform a more thorough search, albeit one which might be more susceptible to overfitting, whereas the evolutionary ensemble approach introduced here uses evolutionary pressure as the only learning technique.

Methodology

Several of the studies described in Section 4.4.3 were assessed on only two benchmark data sets, one of which being the Australian Credit Card problem which was also included in this study. Therefore, a comparison can be made by comparing the results of the ensembles produced in this chapter on this data set with those reported in the literature. However, as these comparison techniques have used accuracy as the evaluation criteria rather than AUC as employed in this work, along with a ten-fold cross-validation resampling profile rather than the repeated cross-validation strategy employed throughout this chapter, adjustments need to be made to the experimental setup to ensure a fair comparison.

Instead of tuning the ensemble parameters for each run, the optimal settings as discovered

Table 5.14: Evolved ensemble algorithms included in the comparison

Algorithm	Study
EENCL	Liu et al. (2000)
MPANN1	Abbass (2003)
MPANN2	Abbass (2003)
DIVACE	Chandra & Yao (2006*a*)
DIVACEII	Chandra & Yao (2006*b*)
DIVACEII-PFC	Chandra & Yao (2006*b*)

previously were implemented for this comparison, namely expression tree base classifiers evolved using DC with the *features* similarity measure, from which the ensemble was built from the twenty most accurate members. The evolved weighted average technique was shown to prove advantageous over the evolved binary average method when optimising for accuracy as opposed to AUC. To report ensemble accuracy, models were trained using AUC as the fitness value guiding the evolutionary learning, while reporting the test set accuracy of the individual using the optimum threshold as calculated on the training set.

Chandra & Yao (2006*b*) details the results of several evolved ensemble algorithms (summarised in Table 5.14) on the Australian Credit data set in terms of their mean cross-validated accuracy with confidence intervals. It is not specified whether the normal or *t* distribution were used to calculate these confidence intervals, however, manual calculation of the intervals using the sample standard deviations supplied in Table 2 in Chandra & Yao (2006*a*) indicates that they have been derived from the normal distribution (albeit with small rounding errors), despite the t-distribution offering a less biased estimate due to the small sample size of ten repetitions. Along with the evolved ensemble techniques summarised in Table 5.14, two statistical ensemble approaches have also been included in the comparison, random forests and AdaBoost. This is motivated by the unexpectedly poor performance of AdaBoost indicated in Table 7 in Chandra & Yao (2006*b*), where it is less accurate on average than all the evolved ensemble techniques, and even the CART decision tree algorithm. These are labelled as *rf* and *ada* respectively in the following analysis. Note that while there are regularised versions of DIVACE-II summarised in Chandra & Yao (2006*b*), the versions included in this comparison are the unregularised versions which perform better. The single objective evolutionary ensemble building technique described in this paper has been labelled *SO* in the following comparison.

Results

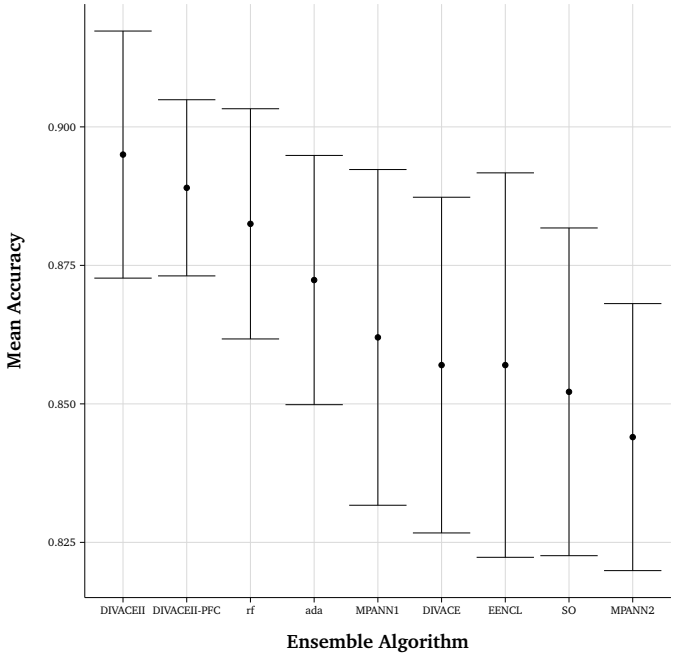


Figure 5.15: Comparison of evolved ensemble algorithms on the Australian Credit problem

The accuracies of the evolved ensembles across ten folds of cross-validation are displayed in Figure 5.15 along with their corresponding 95% confidence intervals (calculated using the standard distribution). There are several immediate findings, first, aside from both formulations of DIVACEII, the evolved ensemble algorithms tend to have relatively similar accuracies. This highlights the ability of the standard single objective EA with niching to compete with more complex MOEA methods, which explicitly reward diverse individuals at the fitness evaluation level, despite the increased computational complexity of these latter approaches which also train individuals using backpropagation at each generation. It must also be taken into account that the only form of model selection for this comparison for the *SO* technique was selecting between one of two voting strategies; it is likely that if all the ensemble building configurations described earlier in this chapter were considered the accuracy would be higher.

The statistical ensemble techniques of random forests and AdaBoost perform very well, with a considerable improvement in accuracy over the majority of the evolved ensemble methods, although there is no evidence of statistical significance in these results, that is most likely due to the small sample size considered here. Furthermore, the mean accuracy of AdaBoost (as implemented in the *ada* R package), shown here to be equal to 0.872 is significantly higher

than the value of 0.843 cited in Chandra & Yao (2006*b*).

Perhaps the most surprising result is that both formulations of the DIVACE-II algorithm are considerably more accurate than not only the remainder of the evolved ensemble approaches, but also the state of the art statistical ensemble algorithms. However, a potential explanation for these results is provided in Table 3 of Chandra & Yao (2006*b*). According to this table, all four formulations of DIVACE-II—with both the NCL and Pairwise Failure Crediting (PFC) diversity penalty functions, with and without regularisation—are on average more accurate on the test set than on the training data. Typically, learning algorithms are considerably more accurate on the data that they are fitted to than on previously unseen data which may not present the same signals evident in the training data, indeed it is demonstrated in Table 5.13 that the evolved ensembles in this work are around 7% less accurate on the test folds. While a model can score more highly on the test fold than on the training folds due to chance, for example the patterns present in the test set could be considerably further away from the discrimination boundary between the two classes, it is highly unlikely to develop a learning algorithm which is on average more accurate on the unseen data. Without a reasonable explanation for this behaviour, it is hard to justify comparing other learning algorithms to DIVACE-II.

5.9.2 Traditional Statistical Learning Algorithms

Forming ensembles from EA populations has been demonstrated to produce more accurate classification models than selecting the single fittest individual overall, particularly when combined with an effective voting strategy. The resulting ensembles were shown to compare competitively to other evolved ensemble learning procedures from the literature, typically employing explicit similarity penalties in the fitness evaluation stage, and often training the constituent ANNs at each generation, thereby providing two sources learning. However, on the Australian Credit data set, this niching approach to ensemble building was shown to perform worse than the established statistical ensemble learning algorithms of random forests and AdaBoost. This section provides a thorough comparison between the evolved ensembles and a number of statistical modeling approaches on a large range of data sets to place these results into the context of the wider field of statistical predictive modeling. Table 5.15 displays the details of ten commonly used classification algorithms—including three ensemble techniques: AdaBoost, Random Forests, and boosted Decision Trees—which will be compared against the evolved ensembles.

Table 5.15: Statistical learning algorithms included in the comparison with evolved ensembles

Model	Abbreviation	Ensemble
AdaBoost	ada	Yes
Boosted decision tree	C5.0	Yes
Unboosted decision tree	C5.0Tree	No
Logistic Regression	log	No
k-Nearest Neighbours	knn	No
Backpropagation trained MLP	mlp	No
Random Forest	rf	Yes
Linear SVM	svmLinear	No
Polynomial SVM	svmPoly	No
Radial SVM	svmRadial	No

Methodology

These learning algorithms were implemented using the Caret package for the R programming language (Kuhn, 2015), and were used to train models on the same ten data sets with the equivalent 10 x 10-fold cross-validation resamples used by the evolved ensembles to provide a matched experimental setup. Many of these algorithms contain hyper-parameters governing model behaviour, which were optimised by sweeping over the default values provided by Caret on each of the cross-validation splits, selecting the combination with the highest average accuracy to represent the algorithm. Likewise, the single most accurate combination of evolved ensemble parameters was discovered to be using GP base classifiers, evolved using the *standard* breeding strategy, from which the ensemble members were picked using the elitist approach with a size of fifty and combined using the evolved average voting technique. Interestingly, these parameters were not the combination of the most individually accurate options as observed previously.

Results

The overall validation set AUC scores of these learning algorithms with their most accurate hyper-parameters settings, across all 1000 validation folds have been plotted in Figure 5.16, alongside the highest scoring combination of ensemble parameters from the EA (highlighted in red for ease of comparison).

The results demonstrate that overall, evolving ensembles performs relatively well compared to established predictive modeling algorithms. It is more accurate than simpler learning techniques such as k-Nearest Neighbours (kNN) and linear Support Vector Machines (SVMs),

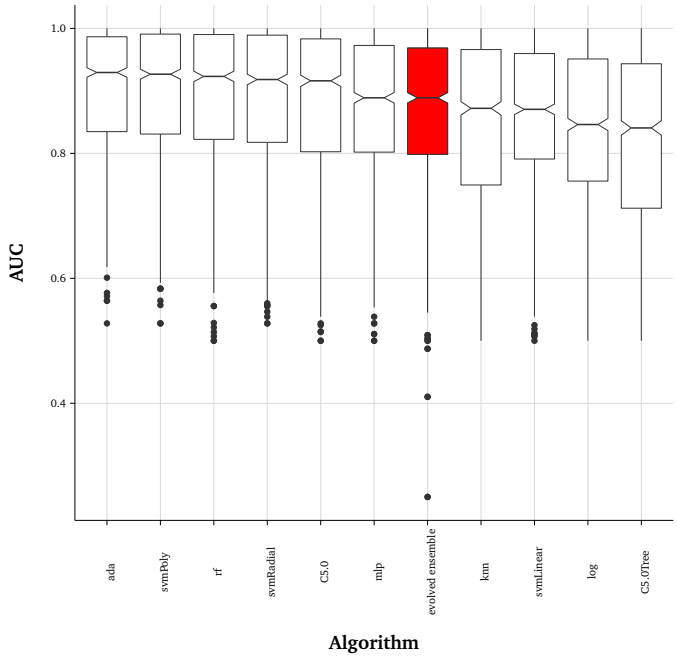


Figure 5.16: Comparison of evolved ensembles with standard modeling techniques

but is less accurate than the three ensemble methods, in addition to the more complex SVM kernels—radial and polynomial. These later algorithms are the current gold standard for classification, scoring higher would have been extremely unexpected. While random forests and boosted trees typically comprise thousands of trees in their ensembles, the proposed evolved ensembles have been shown to function well with just twenty members. This shows that it is easier to manually inspect the produced evolved ensembles than random forests, even if the base evolved trees are just as complex as their decision tree counterparts in random forests. The success of these algorithms using a large number of base classifiers indicates that there is scope to investigate evolving ensembles of similar size, potentially in a large population with some means of subsetting the training set to ensure diverse individuals.

The five most accurate learning algorithms perform very similarly, with an observable decrement in performance separating the following group of models, which includes the evolved ensembles. If the evolved ensembles approach was able to break into this high performing set of models, then it is likely it would receive attention from the wider machine learning community. It is interesting to note that despite comprising an additional source of stochastic behaviour to traditional deterministic learning algorithms, the evolved ensembles maintain comparable levels of variation. Further investigation into reducing overfitting by employing controls during the ensemble training process could further aid ensemble accuracy.

5.10 Conclusions

Multiple aspects of ensemble formation using Evolutionary Algorithms have been investigated in this chapter, including: diversity preservation methods incorporated during the initial training process, selection strategies used to build the combined model, and different approaches to the voting method. The use of niching breeding methods have been highlighted as aiding ensemble accuracy albeit with certain caveats: for example, the particular operator must not overly focus on the explorative side of the elitism vs exploration trade-off, as this can lead to a very diverse population at the expense of accuracy. This was highlighted by the relatively weak performance of PC and RTS. Methods which incorporate controls for keeping highly fit individuals, such as DC and SCEA fared much better. A selection strategy that solely picked individuals based on their fitness was shown to be advantageous over other techniques which measured diversity, however, this was based on the assumption of an existing diverse classifier pool, which could be achieved with the aforementioned niching strategies. Training the ensemble by means of a secondary evolution stage was shown to significantly aid overall accuracy, providing the greatest impact of the factors investigated, with the choice of niching algorithm second. Interestingly, the base classifier architecture had little effect on overall ensemble accuracy, indicating that the learning algorithm is more important than the model representation and is where the majority of optimisation efforts should be focused.

While these aspects of evolving evolutionary learning have been investigated previously in the literature, they have typically been investigated individually and in the context of complex MOEAs. The work described in this chapter has demonstrated that many of the same concerns are true for single objective EAs, and indicate that this approach of indirectly optimising for diversity via the use of niching algorithms can build ensembles more accurate than their fittest members without necessitating the explicit dual optimisation of both accuracy and diversity seen when using MOEAs. Considering numerous factors in ensemble building at the same time has produced a thorough survey of different ensemble parameters across a large range of benchmarks, which has not been investigated on this scale previously. The

Evolving ensembles using a traditional single objective EA and incorporating diversity at the breeding stage via the use of niching methods was shown to perform competitively with more complex MOEAs algorithms, which benefit from training the constituent ANNs through both evolutionary pressure but also backpropagation, thereby requiring considerable computational expense. Since building a model for an applied situation requires calibrating various hyper-parameters, having an algorithm which can train a model quickly is a large

benefit, and so this niching technique shows promise for future development. However, more refinement is required before it can outperform the current best in field methods of AdaBoost, non-linear SVMs, and random forests. The success of boosted trees and random forests, which typically comprise thousands of base classifiers compared to less than one hundred in the evolved ensembles, suggest the possibility of increasing the size of the evolved ensembles to further increase classification accuracy. This could be achieved by substantially increasing the population size and using a means of ensuring that niches are trained on differing subsets of the training data, such as co-evolution, to preserve diversity in a similar manner to these established techniques.

The fact that evolved ensembles are most beneficial over individual classifiers on smaller data sets highlights their suitability for use with real world medical data, where collecting observations can prove logistically challenging and time consuming. The evolved ensembles developed during this chapter will therefore be applied to diagnosing PD patients, as will be investigated in the following chapter.

Chapter 6

Applying Ensembles to Parkinson's Disease Movement Data

Contents

6.1 Introduction

In order to examine the physiology of Parkinson's Disease (PD) in greater detail than previously managed, a relatively large number of both PD patients and healthy control subjects had motion data recorded by high resolution non-invasive sensors while performing a simple movement task—finger tapping. A data set was subsequently formed and used for three purposes: (i) a clinically motivated investigative study to establish the effectiveness of an area of the Movement Disorder Society sponsored revision of Unified Parkinson's Disease Rating Scale (MDS-UPDRS), the findings are detailed in Appendix B as it lies outside the main narrative of this thesis; (ii) developing models to differentiate between healthy controls and patients using the previously discussed evolved ensembles; (iii) the final approach modeled cognitive impairment as a function of movement disorder. Under expert clinical guidance, seven measures of bradykinesia were derived during the initial investigation into the MDS-UPDRS efficacy and subsequently used for the later two modeling tasks. This chapter describes the latter two of these aspects in turn, beginning by detailing the data recording methodology.

6.2 Movement Data Collection

6.2.1 Test Subjects

Three international healthcare centres—Leeds General Infirmary (LGI) (Leeds, UK); the Memory and Aging Center (MAC) at the University of California, San Francisco (UCSF) (California, USA); and the San Francisco Veterans Association (SFVA) (California, USA)—participated during the data recording phase resulting in a mix of patients recorded from a variety of demographic backgrounds, with different experiences with PD. Two separate periods of data recording were run at LGI to analyse different aspects of PD; both of these trials recruited similar numbers of test subjects and have been labelled *LGI1* and *LGI2* in chronological order for ease of interpretation. The recordings at the MAC and SFVA took place during the same time frame under the same direction, and so have been grouped together into a single collection named *UCSF*. Details of the centres and the number of recordings at each are found in Table 6.1. Patients recorded at LGI were observed while under the effects of Levodopa, while patients observed at UCSF were recorded twice: once while on medication and again after it had worn off. For the sake of consistency, only data recorded while a patient

Table 6.1: Details of the test subjects observed at each centre

Centre	Identifier	Control Subjects	PD Patients
LGI (first study)	*LGI1*	41	49
LGI (second study)	*LGI2*	29	58
MAC and SFVA combined	*UCSF*	25	39
Combined	*all*	95	146

was under the effects of medication was analysed during this work. While a diagnostic tool would aim to identify cases of un-diagnosed PD, and thereby in people not currently taking Levodopa, it is a more challenging task to discriminate between healthy controls and PD patients taking medication to mask their symptoms. In addition, it was not possible to obtain movement data from patients off medication at all the centres owing to issues associated with obtaining ethical approval.

In addition to the finger tapping tasks, patient meta-data was recorded during the visits. This included information such as the patient's age; the length of time since the PD diagnosis was made; scores on the MDS-UPDRS assessment criteria on both hands; current Levodopa dosage (if applicable); as well as three cognitive assessments comprising the Montreal Cognitive Assessment (MoCA), Clinical Dementia Rating (CDR), and the Mini-Mental State Examination (MMSE).

6.2.2 Finger Tapping Protocol

The movement task consisted of a simple repetitive motion whereby a person taps their index finger against their thumb for a predetermined duration of time, known as *finger tapping*; it is included in the MDS-UPDRS diagnostic criteria as item 3.4. Despite its simplicity, finger tapping provides an insight into a patient's condition by highlighting the cardinal symptom of PD known as bradykinesia, exhibited by slowness of movement, hesitations, and a reduced range of motion. The periodic nature of the task also allows for the observation of the sequence effect—a decrement in amplitude and speed of repetitive movements over a short period of time, more severe than standard physical fatigue. More details of these impairments are found in Section 2.2.2.

Test subjects were recorded performing a subtly modified version of the finger tapping protocol laid out in the MDS-UPDRS (Goetz et al., 2008), which contains the following instructions (emphasis added):

> "Instruct the patient to tap the index finger on the thumb 10 times as quickly AND as big as possible."

The task was performed twice, once with the dominant hand before being repeated with the non-dominant hand. However, the recording lasted thirty seconds rather than the ten tap limit enforced by the MDS-UPDRS, thereby allowing more time for bradykinesia symptoms to be exhibited. Test administrators were instructed to mark the patient's level of movement disorder on a scale of *Normal*, *Mild*, *Moderate*, and *Severe* with the MDS-UPDRS item for finger tapping providing guidance on how to form an assessment:

> "Rate each side separately, evaluating speed, amplitude, hesitations, halts and decrementing amplitude."

By obtaining movement data, each of these aspects can be assessed objectively and in greater detail than is currently provided by the MDS-UPDRS.

6.2.3 Equipment

The finger tapping cycles were recorded by a pair of non-invasive Electro-Magnetic (EM) sensors attached to the test subject's index finger and thumb as shown in Figure 6.1. These lightweight and small transducers recorded at a frequency of 60Hz with a resolution of 1.52mm, allowing for a detailed measurement of any movement disorders. To provide a measure of position, an EM signal was transmitted by a separate source unit placed approximately 50cm away, and received by the two sensors which subsequently relayed their relative position to an attached central processing unit. Movement was recorded in six degrees of freedom, however, only positional data was used in this study.

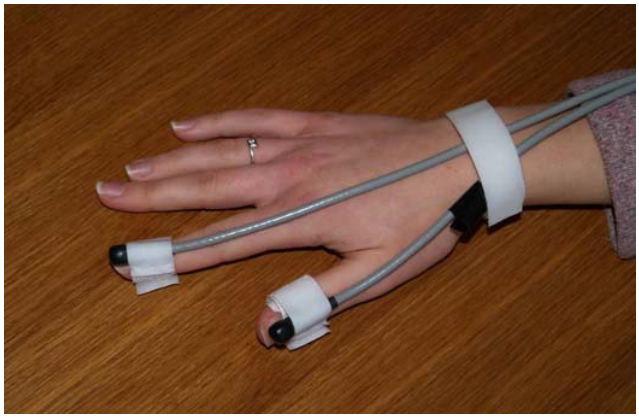


Figure 6.1: The position sensors used in the study were lightweight and unintrusive when performing movement tasks. Image was originally used in Edgar (2007)

6.2.4 Data Processing

The raw data comprised (x_{ij}, y_{ij}, z_{ij}) coordinate points for sensors $i \in \{1, 2\}$ and time steps $j = 1, 2, \ldots, N$, measuring the distance from the source unit. The Euclidean distance between the two sensors was calculated at each sample to obtain separation points (x_{sj}, y_{sj}, z_{sj}), which were subsequently converted to a waveform. This separation signal was smoothed using a low-pass Butterworth filter with ω_c set at 5Hz, with the aim of producing a clean sinusoidal separation trace as shown in Figure 6.2.

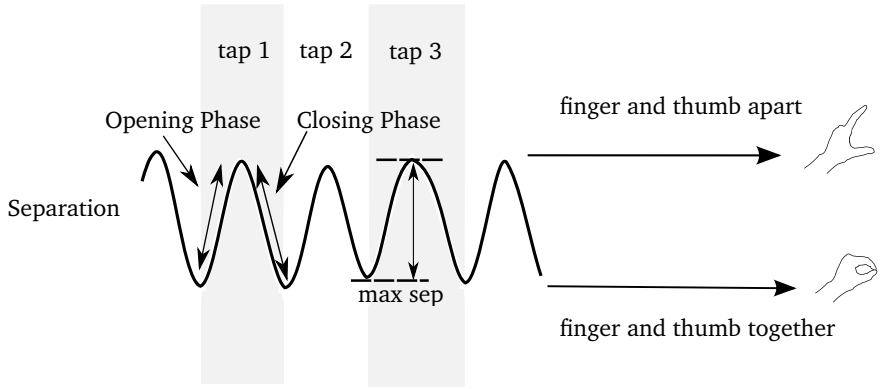


Figure 6.2: The finger tapping movement produces a sinusoidal wave when processed

From the separation signal, finger tapping velocity and acceleration were calculated as the first and second derivatives respectively. By locating local maxima and minima in the separation waveform, the data samples were segmented into tapping cycles in which several identifying features could be calculated, as shown in Figure 6.3. An initial transient behaviour can be observed, whereby the separation signal is not immediately identifiable as representing finger tapping behaviour, due to reaction time to the cue to start the motion. To alleviate any discrepancies caused from misidentifying any taps during this period, the first tap cycle from each recording is discarded.

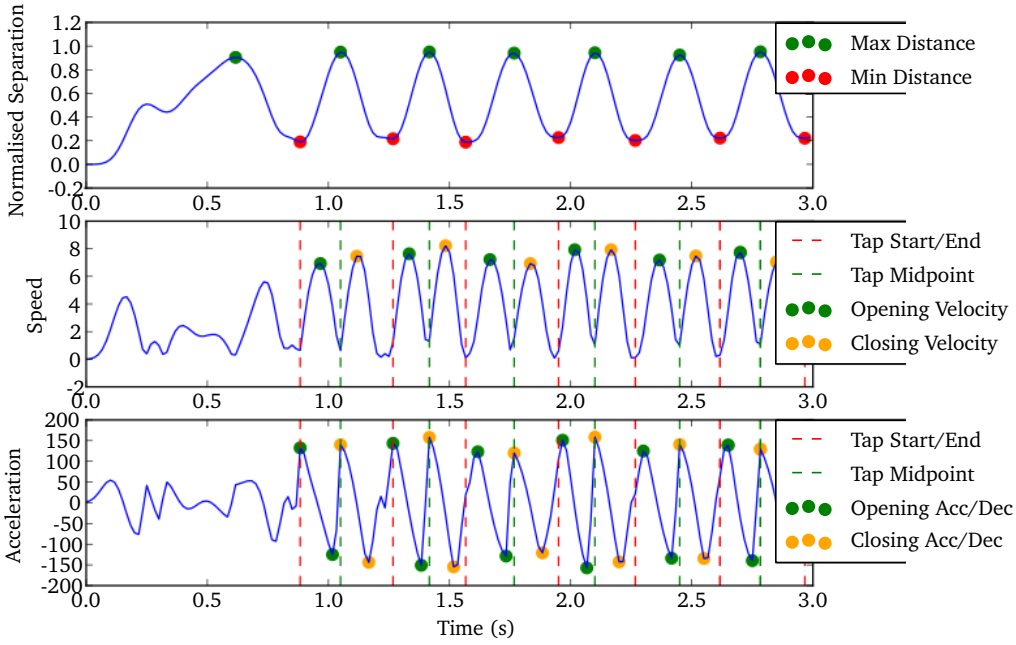


Figure 6.3: Identifying measures could be derived for each tap cycle from the separation trace

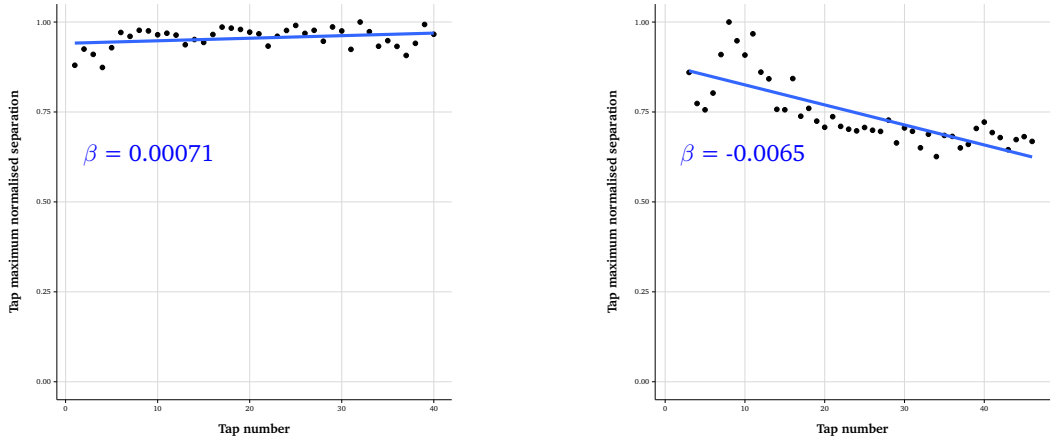
6.3 Modeling Bradykinesia

As discussed in Section 2.2.2, an early and accurate diagnosis of PD is essential to ensuring the most effective clinical pathway is followed. By modeling positional data recorded at the three international centres in the form of movement disorder markers extracted using expert medical knowledge, the ability of evolved ensemble classifiers to differentiate between PD disease and a healthy control subject will be investigated. Seven different summary measures were extracted from the raw movement data under clinical guidance to assess the following components of Parkinsonian movement disorders:

- The sequence effect (progressive decline in amplitude over a short time duration)
- True bradykinesia (general slowness of movement)
- Akinesia (hesitation)
- Hypokinesia (decrement in movement amplitude)

As the MDS-UPDRS finger tapping criteria only instructs clinicians to assess amplitude and speed, summary measures were formed from these corresponding raw waveforms and not from acceleration. At each tap cycle, the maximum separation and velocity were calculated to

obtain a local measure of amplitude and speed, which were aggregated in various ways to produce a single value summarising the entire recording. The mean tap maximum separation and speed were calculated to represent hypokinesia and true bradykinesia respectively, while akinesia was quantified by the coefficient of variation of these tap values. However, a more involved method was required to obtain a measure of the sequence effect. A simple linear regression model was fitted to the tap separation scores as a function of the tap number, with the resultant slope used to measure any progressive decrement in amplitude and thereby the sequence effect, as demonstrated in Figure 6.4. This process was also repeated for speed to obtain a secondary measure of the sequence effect. The final extracted feature was the tapping frequency.



(a) A healthy control (b) A PD patient

Figure 6.4: Examples of fatigue in healthy test subjects and controls

In total seven features were extracted from the raw data and are summarised in Table 6.2, along with labels used to refer to them by for the remainder of the chapter. The following sections discuss modeling PD from these bradykinesia markers using the evolved ensemble setup derived in Chapter 5, with two main objectives:

1. Establish whether evolved ensembles aid identification of PD patients from simple movement data

2. Determine if a PD patient's level of cognitive impairment can be determined from their motor symptoms

Once a model has been fitted it is possible to analyse it to understand which movement components are being assessed to form the diagnosis, providing useful clinical feedback.

Table 6.2: Bradykinesia features extracted from the raw data

Feature	Identifier	Tap measure	Aggregation function
Tap frequency	*freq*	NA	NA
Average tap separation	*mean.max.sep*	Peak separation	Mean
Average tap speed	*mean.max.speed*	Peak speed	Mean
Variability in amplitude	*cov.max.sep*	Peak separation	Coefficient of variation
Variability in speed	*cov.max.speed*	Peak speed	Coefficient of variation
Decrementing amplitude	*fatigue.sep*	Peak separation	Regression line gradient
Decrementing speed	*fatigue.speed*	Peak speed	Regression line gradient

6.4 Identifying Parkinson's Disease

6.4.1 Overview

The first set of experiments investigated the diagnostic ability of evolved ensembles trained on the seven summary bradykinesia features described in Table 6.2. The aim was to establish not only whether PD could be distinguished from healthy controls, but also if the discriminatory ability varied across the centres; a robust and accurate model of a disease should be able to provide similar results no matter where it is deployed. In addition, the resultant models are analysed to determine which bradykinesia components are useful for differentiating between healthy controls and PD patients, and compared with the current MDS-UPDRS guidelines.

6.4.2 Methodology

Evolved ensembles were trained on four different data sets, representing each of the three international medical centres individually, and one combined group, with the number of healthy controls and PD patients in each detailed in Table 6.1. The task was a dichotomous binary classification one, with the objective of identifying whether a data sample belongs to a healthy control or a patient. Despite previous results indicating that classifying data from the dominant hand only has greater predictive power (Lacy et al., 2013), the data sets used in this study comprised recordings from both hands to increase training set size to aid generalisation. The previously derived most effective ensemble building parameters were used to form ensembles of classifiers, consisting of:

- Genetic Programming (GP) expression tree base classifiers
- Deterministic Crowding (DC) breeding operator using the *features* similarity measure

- Ensemble sizes of twenty selected by elitism
- The evolved average voting method

The data was passed into the model in the same manner as for the classifiers analysing the University of California Irvine (UCI) benchmarks (Chapter 5), whereby the expression tree input set consisted of the data attributes with one being chosen at random for each new input leaf node and a point mutation operator which selected a new random feature from the set. The feature vector comprised the seven movement summary measures, which were the bradykinesia measures extracted from the raw data as summarised in Table 6.2 along with the labels used to refer to them throughout the remainder of this chapter. To provide a thorough overview of the evolved classifier's generalising ability, the data was resampled using thirty repeats of ten-fold cross-validation.

6.4.3 Results

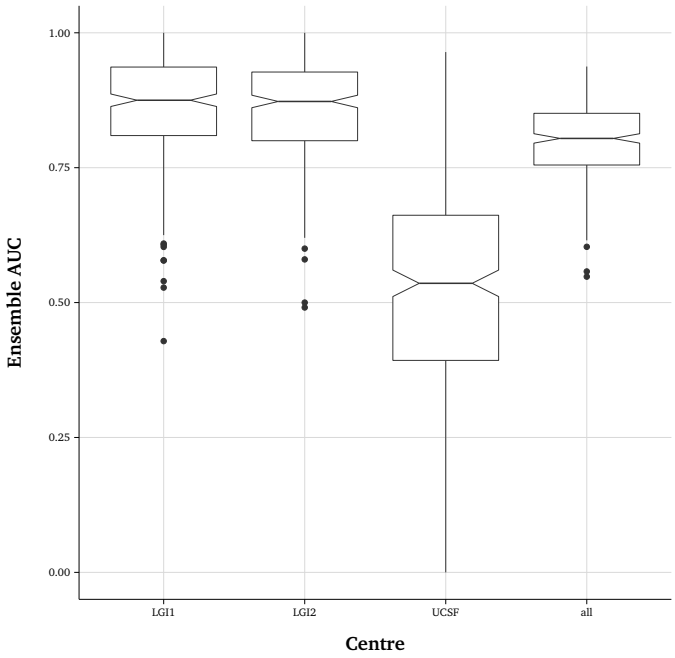


Figure 6.5: Ensemble AUC scores from the four recording locations

Figure 6.5 shows the overall classification ability of models formed on PD data from the four different recording locations across the three hundred validation folds for each centre. The location where the movement tasks were recorded has a significant effect upon the discriminatory ability of the evolved classifiers; the two LGI data sets have extremely similar

median AUC values—0.875 and 0.873—while this is reduced to 0.536 on the data recorded at *UCSF*, barely improving upon a random guess. This highlights the challenges faced when collecting real data on a global scale, as subtle differences in the data collection methodology can have significant effects on the recorded values. These differences could consist of variations in instructions given to the participant, slight discrepancies in the equipment setup, or varying environmental factors. However, if the clinical methodology is preserved, then generalisation can be achieved, as evidenced by the similarity in performance of models trained on both the data sets recorded at LGI, despite the data being collected several years apart by different administrators.

For example, instructing a test subject to focus on maintaining a maximum speed throughout the recording could prove disadvantageous as movement speed is less discriminatory between healthy subjects and PD patients, owing to the fact that Levodopa is more effective at improving movement speed than amplitude (see Section 2.2.2). An additional explanation for this discrepancy in diagnostic ability at *UCSF* stems from the high precision of the EM position sensors. While having a high resolution is beneficial overall, it can also have its drawbacks. For example, any differences in the placement of the sensors on the thumb and index finger—or if the sensors slip during recording—will have a large impact on the baseline of the resulting separation waveform.

Overall, however, the ensembles are able to diagnose PD with a relatively high level of precision, scoring a median AUC of 0.804 on the *all* data set which includes the noise containing recordings from UCSF. These results highlight the potential for using movement data as a diagnostic aid for PD, although the importance of ensuring a consistent experimental setup is also demonstrated. While the classifier accuracy on the *all* data set is significantly lower than that from either of the LGI data sets, it is still within a range indicating a useful discriminatory power, thereby demonstrating that combining the raw movement data—potentially recorded in different circumstances—can help to mask any noise encountered. This theory is explored in greater detail in Section 6.4.5.

6.4.4 Use of Bradykinesia Features

Using an ensemble of expression trees facilitates an insight into how predictions are formed, as these base classifiers incorporate an inherent feature selection process; evolution will select data attributes for the tree associated with accurate models. By counting usage rates of each attribute, an overview of which movement components contribute to PD diagnosis can be

produced. This is in contrast to another Computational Intelligence (CI) technique which is frequently used as a classifier, Artificial Neural Networks (ANNs), which are commonly considered as black boxes with little understanding of how the prediction is being made or which attributes are most discriminatory due to the high number of interactions between inputs. Figure 6.6 shows the frequency of each of the seven movement disorder markers in the most accurate evolved ensemble on each data fold for each recording location data set.

Overall, *mean.max.sep*, *mean.max.speed*, *cov.max.speed*, and *fatigue.speed* were strong indicators of PD as these attributes were most frequently selected by evolutionary process. It is interesting to observe significant differences in feature usage between clinics, for example *fatigue.speed* was used very frequently when modeling the *LGI2*, *UCSF*, and *all* data sets but was the least selected feature in the *LGI1* location. Likewise, *mean.max.sep* was observed in a large proportion of models from all data sets except *LGI2*, while every expression tree in every ensemble on every fold on the *LGI2* data set incorporated the *mean.max.speed* attribute.

Another difference between the data sets is the emphasis put on amplitude versus separation characteristics. The *LGI2* models used almost exclusively measures of speed, which, while incorporated by classifiers on *LGI1* and *all*, were augmented with amplitude features (commonly in the form of *mean.max.sep*). These observed differences could indicate either discrepancies between the PD severity of the patients recruited at each location, or variability in the assessment instructions. Certain locations may have focused the subjects on producing as fast taps as possible, allowing for indicators of PD to become more pronounced.

The use of evolved ensembles for diagnosing PD has produced not only accurate classification models, but also enabled a simple overview into how these predictions are being formed, indicating to clinical experts which movement components are indicative of PD.

6.4.5 Improving Classification Ability on *UCSF* data set

The low cross-validation accuracy of classifiers modeling the *UCSF* data set is indicative of a poorly fit model with a large error term. Recalling the bias-variance decomposition of error previously discussed in Section 3.3, the total error is a composite of three parts: intrinsic noise, bias, and error due to variance. It is likely that the intrinsic noise for this data set comprises a significant portion of the overall error term, but it remains to be seen how the remaining error is distributed between bias and variance. It is typically easier to reduce error due to variance than from bias, as this involves producing a model with better generalising capabilities, which can be achieved by increasing the number of training samples,

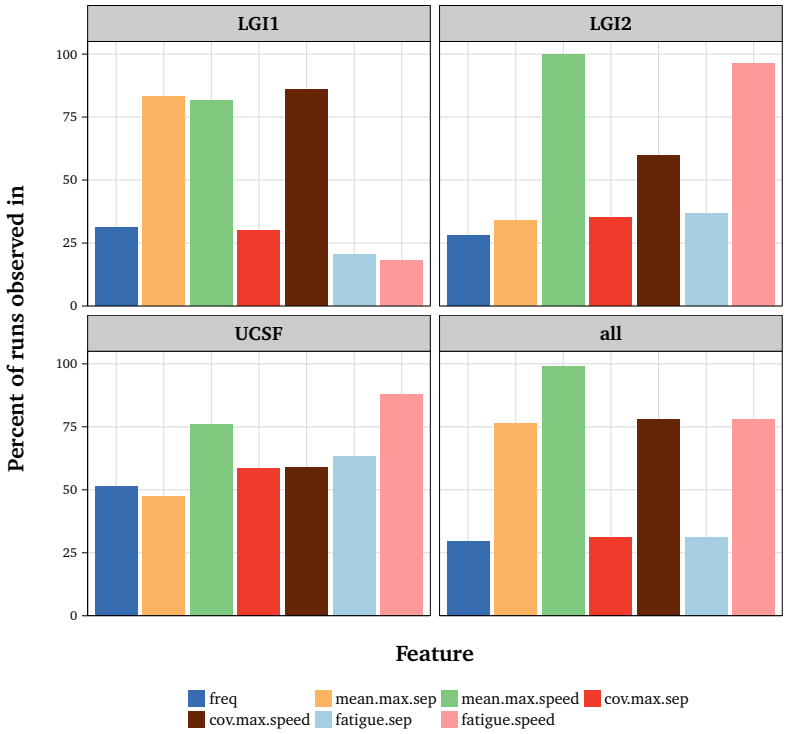


Figure 6.6: Bradykinesia feature usage rates across all the recording locations

or reducing the model complexity. If the intrinsic noise is large, then reducing the bias error component can be a challenging process, generally requiring a more complex model capable of identifying non-linear patterns in the data. However, the evolved ensembles are already forming highly non-linear discrimination functions through a thorough search and are unlikely to find significant improvement this way. An alternative approach to reducing bias error is to include more discriminatory features in the data set, however, for this work evaluating the efficacy of such ensemble approaches instead of being a direct application, this method will not be considered.

To establish the size of the variance error, the measure of overfitting derived in Section 5.7.5 was calculated at each recording location, as $\frac{AUC_{tr}-AUC_{va}}{AUC_{tr}}$, with these terms representing the training and validation set AUC scores respectively. The resulting overtraining summary scores are displayed in Table 6.3, and immediately indicate a potential source of the low accuracy on the *UCSF* data set. The AUC on a training set of data recorded at this location is on average 328% greater than the corresponding validation fold score, a far greater difference than encountered by the remaining data sets. Overfitting could be occurring as a result of the size of the data sets, with *UCSF* comprising 64 data instances, compared to 90 and 87 for

LGI1 and *LGI2* respectively. However, there is most likely an additional compounding factor, as the difference in data set size is not large enough to warrant such contrasting results alone. The overfitting values indicate that the model accuracy can be improved on the *UCSF* data set by reducing the overfitting to the training set, and thus decreasing the contribution of the variance portion of the total error term.

Table 6.3: Ensemble overtraining on each data set

Recording location	Mean percent AUC overfitted
LGI1	5.56
LGI2	5.12
UCSF	32.10
all	3.65

Standard methods of achieving this include adding regularising terms, increasing the size of the training set, and reducing model complexity. Out of these approaches, increasing the number of training samples is the easiest to manage while still maintaining a fair comparison with the other data sets. While all the data recorded at *UCSF* has been exhausted, the *all* combined data set provides an extended number of samples. By evaluating the efficacy of classifiers trained on the *all* data at predicting unknown patterns taken from *UCSF*, not only can overfitting on this data set be reduced, but, if successful, it can provide useful feedback about combining data from multiple sources.

The methodology for this investigation comprised evaluating the ensemble trained on each set of training folds of the *all* data set, on the samples in the validation fold which were recorded at *UCSF*, thereby providing an estimate of generalising ability to unseen data. This process was repeated on all of the 30 x 10 cross-validation resamples, and then compared to the cross-validation accuracy of the classifiers solely trained on the *UCSF* data. While the specific validation folds are not matched, the large number of resamples reduces the variance from the resampling of the data set enabling a fair comparison. The results of these simulations are shown in Figure 6.7.

The plot visually shows that the validation set AUC scores of classifiers fitted on the *all* training samples are greater than those trained exclusively on the *UCSF* training data; the median AUC obtained on *UCSF* is 0.536, while training the ensemble on the full data set results in a median AUC of 0.600. Thus, increasing the training size, even by adding in training samples which may contain different properties—such as less intrinsic noise and having been recorded under different circumstances—is shown to have a quantitative improvement in this instance. A

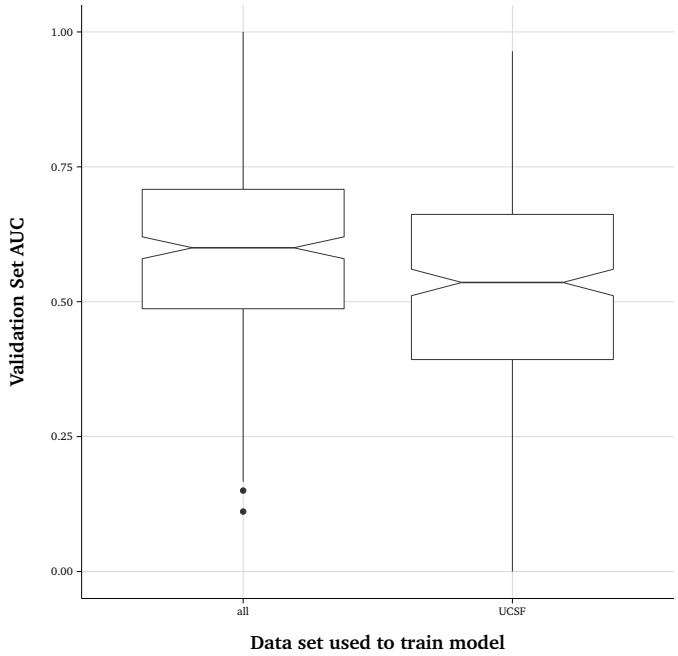


Figure 6.7: Comparison of models trained on different data sets and evaluated on *UCSF* samples

Mann-Whitney U-test (employed instead of the Wilcoxon signed-rank test as the data instances are not matched) produces a p-value < 0.1, providing statistical significance to these findings.

This procedure was repeated for the *LGI1* and *LGI2* data sets to establish whether any increase in model accuracy could be gained using the data trained from the data across all the centres, however the results were inconclusive with no statistical significance found—p-values of 0.625 and 0.138 were calculated for *LGI1* and *LGI2* respectively. These finding suggests that the variance component of the model error on these data sets was already relatively minimised, with the majority of the error arriving from intrinsic noise and bias. However, since there was no net loss in accuracy by using the classifiers trained on the *all* data set, recommendations for applied modeling in the medical community would include fitting the final model to as much of the data set as possible, no matter how varied.

6.4.6 Discussion

The results highlight several important considerations to be made when recording movement data using sensitive equipment in different clinical environments. First, care must be taken to ensure that the experimental setup is as uniform as possible across sites, else large amounts of noise can exist in the data, leading to classifiers forming less accurate models than expected.

However, the fact that the data was less discriminatory from one centre can prove useful in additional ways, as the source of this noise can be an indicator of how PD manifests itself in comparison to healthy subjects. For example, if it were discovered that the participants at UCSF were instructed to focus on completing as many finger tapping cycles as possible within the time limit with less regard for achieving maximum separation, then it could be deduced that the amplitude element of bradykinesia is more defining than the reduced speed.

Evolved expression tree classifiers implement an inherent form of feature selection, as the individuals using the most discriminatory attributes are more likely to be bred through subsequent generations. The use of parsimony pressure prevents trees from growing too deeply and including every data feature, instead only being able to select the optimal inputs. Tallying the usage of each data feature in the final ensemble, which have been selected by simulated evolution due to their presence in the fittest classifiers, allows for an insight into which movement components are most discriminatory between PD patients and healthy test subjects. This is extremely valuable information from the perspective of the medical clinicians, who can compare the means by which they form a diagnosis—such as the Unified Parkinson's Disease Rating Scale (UPDRS)—to those derived from an objective system. Furthermore, this analysis indicates that different components of the data are being used to classify across the different clinics, highlighting differences in the experimental methodology.

The finding that the poor classification accuracy on the *UCSF* data could be significantly improved by employing ensembles trained on the combined data set has important repercussions both for this work and future multi-centre studies. Good practice will incorporate as great of a variety of data as possible, even if there are large amounts of intrinsic noise present, as this will enable more general patterns to be discovered, thereby reducing the variance component of the error term.

6.4.7 Comparison with Current Clinical Practice

It is impossible to perform a direct comparison of the diagnostic accuracy of this modeling technique and that from clinical neurologists for a number of reasons, for example: the subjects who were included in this study represent a different subset of the population to those would present themselves to general practitioners or neurologists for diagnosis, as the PD patients included in this study have already been diagnosed with the disease and the controls have shown no indication of having the condition. However, it is possible to perform a qualitative comparison between this automated approach and current clinical practice for

diagnosing and monitoring PD.

Currently, PD diagnosis begins with a person observing decrements to their motor and/or cognitive faculties. A general practitioner will perform an observational test, primarily looking for indications of the cardinal symptoms of PD, including physical symptoms such as tremor, slowness of movement, and rigidity, along with any evidence of cognitive decline or issues with balance. If required, the person can be referred to a neurologist for an expert opinion. However, as described in Section 2.2.2, up to 24% of all initial diagnoses can be inaccurate (this figure was obtained by comparing diagnosis after a biopsy with the initial diagnosis formed using the UK Parkinson's Disease Society Brain Bank criteria Jankovic (2008); Rajput et al. (1991)). This is due to the similarity in both the physical and neurological symptoms with other conditions. For assessing disease progression, or the impact of medication, a more thorough process is required. As described in Section 2.2.2, the gold standard for rating PD severity is the UPDRS, which consists of self-assessed questionnaires and motor examinations and can take several hours.

Evidently, an accurate and early diagnosis of PD would provide numerous benefits to both patient care and ease the burden on the healthcare provider. Chahine & Stern (2011) review the current researched approaches towards this goal, which can be categorised into clinical, laboratory, and imaging methods, concluding that a combination of all these three techniques will be required to produce the most accurate diagnosis. However, these methods require considerable expense and expertise to perform the tests and run the analysis, while an automated machine learning approach can provide a simple confidence value alongside its predicted diagnosis and only requires a specialised test administrator to record the data.

In contrast to these observational assessments, the movement data recorded in this study was recorded from a single component of the UPDRS motor examination, the finger tapping test. Each hand was assessed for just thirty seconds with minimal setup, requiring simply attaching small position sensors to the index finger and thumb with velcro straps. To obtain a measure of movement abnormality, experts in PD derived a set of seven summary measures which could be simply calculated from the data, as described in Section 6.3. Once the evolved ensembles have been trained to the data, any further diagnosis would be produced almost instantaneously as it would simply require extracting the summary features from the separation signal, before evaluating the outputs from each of the ensemble members and subsequently being aggregated.

The approach described above provides several benefits over the traditional diagnosis route.

For example, it is very quick to run, as it only requires a thirty second recording time in addition to minimal setup of non-invasive recording apparatus. In addition, the test result can be produced instantly and facilitates simple interpretation as it is a scalar value, rather than requiring processing or analysis time as in the case of the blood biomarkers and imaging tests currently researched. While expert human knowledge was required to derive the routines to extract the summary features, it is not required at any other stage of the process, although care will need to be taken to ensure the subject follows the established movement protocol. The equipment overhead costs and the running costs of this approach are substantially less than that required by biomarker sampling or imaging tests. Once the recording equipment has been setup, it would be straightforward to record the subject performing other movement tasks in order to form a composite diagnosis, thereby employing the same methodological approach as ensemble building to combine experts in different areas to provide a more accurate overall prediction. Since the UPDRS movement assessment consists of eighteen movement tasks, this procedure could be simply incorporated into current clinical practice for staging disease progression or employed as a stand-alone diagnostic aid.

However, there are still areas of this predictive modeling approach to PD monitoring that would need to be fine tuned to reach the goal of being a viable alternative to human diagnosis. As observed in Section 6.4.5, the recording of movement data during this simple task is subject to variation across the trials which could be due to several factors, for example geographical or demographic differences in the sample population, differences in the instructions relayed to the test participants, variation in the hardware, or environmental factors causing interference with the sensitive EM equipment. Furthermore, while the goal of this research is to produce an objective diagnostic aid, experts in PD were required to aid in extracting the summary features for this task. Now that the algorithms to calculate these measures are established, any future finger tapping task can be automatically analysed using the evolved ensembles, however, if any additional movement tasks were to be incorporated to provide a composite diagnosis, then further clinical expertise would be required to summarise these tasks into further discrete feature sets.

6.5 Differentiating Cognitive Abilities in Parkinson's Disease Patients

6.5.1 Overview

As discussed in Section 2.2.2, PD affects not only a person's motor skills but also has a deteriorating impact upon cognitive ability, which gradually degrades as the disease progresses. One method of quantifying this cognitive impairment is the Montreal Cognitive Assessment (MoCA), a short test assessing multiple aspects of a patient's cognitive functionality including: short-term memory, visuospatial ability, several aspects of speech, language, attention, and concentration. A score is assigned to the participant ranging from zero to thirty—with smaller scores indicating decrementing cognitive ability—and has been shown to perform well when assessing Mild Cognitive Impairment (MCI) (Freitas et al., 2013). The Clinical Dementia Rating (CDR) is an alternative questionnaire used to stage a patient's cognitive impairment, primarily owing to the impact any deterioration has on daily living. This section investigates how well the evolved ensemble classifiers can differentiate between varying levels of cognitive impairment in PD patients, based purely on their finger tapping recordings.

6.5.2 Methodology

Nearly all the patients recorded at the *LGI2* clinic (four were excluded due to logistical reasons) were also assessed by the MoCA and the CDR to obtain an overview of their mental ability alongside their movement characteristics. A minimum value on the MoCA test corresponds to greater cognitive impairment, while it is the opposite using the CDR scale.

The patients were subset into the following three ordered groups of cognitive impairment:

1. No cognitive impairment
2. Mild Cognitive Impairment (MCI)
3. Parkinson's Disease with Dementia (PDD)

The cohort was firstly thresholded by the MoCA scores into two groups, with the PDD and MCI patients being further differentiated by their CDR score, as one of the defining characteristics of PDD is its strong impact upon daily living. This is detailed in Table 6.4 along with the size of each group.

Table 6.4: How the patients were subset into three groups of cognitive ability

Cognitive ability	Identifier	MoCA score range	CDR score range	Number of patients
No cognitive impairment	*nc*	≥ 26	Unused	22
Mild cognitive impairment	*mci*	< 26	$0 - 0.5$	22
PD with dementia	*pdd*	< 26	≥ 1	10

Rather than form an overall multinomial model attempting to classify each data recording to its correct group, three separate data sets were formed to investigate the discriminatory ability between each pairwise grouping of these classes, allowing for a more detailed insight into the subtle differences in movement disorders between them.

Features from both dominant and non-dominant hand recordings were included in the data sets to effectively double the training set size. The classification models used the same optimal evolved ensemble configuration as determined by the UCI benchmark data sets investigation (Chapter 5) that were used to identify PD from the finger tapping data. Using expression tree base classifiers again allowed for a simple insight into which movement components separate the different cognitive groups. The ability of ensembles to model each data set was evaluated using thirty repeats of ten-fold cross-validation, allowing for a reliable indication of generalisation ability.

6.5.3 Results

The AUC of the evolved ensembles on each of the data sets across the three hundred validation folds are displayed in Figure 6.8. The primary result is the significant difference in classification ability between the different levels of cognitive impairment. Unsurprisingly, the data set that the classifiers achieved the greatest success with was the comparison between no cognitive impairment and PDD, on which the discriminatory success is comparable to that between overall PD patients and healthy controls on data recorded at this centre (Figure 6.5). Interestingly, the two least separable groups were those suffering with MCI and those with no cognitive impairment—having a median AUC of 0.500, equal performance to a random guess—indicating that the difference in motor skills between MCI and PDD is greater. The large range of accuracies can be explained by the relatively small data sets; when using ten-fold cross-validation the validation set is only a tenth of the already small collection of samples, meaning that the presence of a single outlier in the validation fold can have a large impact. However, the largest data set was *nc.vs.mci*, indicating that the lack of model fit was

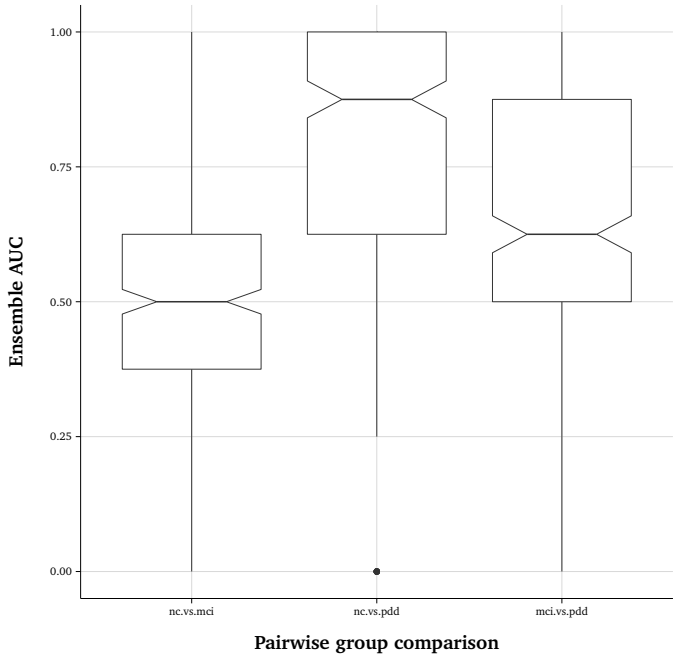


Figure 6.8: Classification abilities between different stages of cognitive impairment

primarily due to a lack of detectable signal in the data, rather than overfitting to the small training set.

6.5.4 Use of Bradykinesia Features

By implementing the same method as Section 6.4.4 of counting the number of times a specific data attribute is used by an ensemble member, an insight is offered into which movement characteristics differentiate the varying levels of cognitive impairment. This was achieved in the same manner as before, whereby the data attribute usage frequency was calculated across every member in the ensemble for all three hundred folds of repeated cross-validation. The results are displayed in Figure 6.9.

The most immediate result is the difference in frequency distributions of the *nc.vs.mci* comparison to the other two data sets, suggesting that patients with no cognitive impairment and those with MCI have more in common than they do with subjects suffering from PDD, reinforcing the severity of the condition. It is also important to remember that the classifiers trained on the *nc.vs.mci* data set were unable to produce models more accurate than random guessing, and so the features used by these classifiers are not necessarily indicative of subtle motor differences between these two groups of PD patients.

Another key finding is the large usage of the *cov.max.speed* bradykinesia measure in two of the

three data sets, where it is the most frequently used measure, despite not being implemented as often in the *nc.vs.mci* models. This indicates that patients with progressed PD have trouble forming regular speed movements, while this difference is less pronounced in those with MCI or no cognitive impairment, suggesting that akinesia is more developed in patients with severe cognitive impairment. This variance in tap speeds could also be due to the *cog-wheel* motion, often found in PD patients. Similarly, the *fatigue.speed* measure, which is closely related to the sequence effect, was frequently used to identify PDD, but was the least selected data attribute in the *nc.vs.mci* comparison. This highlights that the sequence effect is primarily associated with patients who also exhibit high levels of cognitive impairment.

As with the evolved ensembles differentiating between healthy controls and PD patients at a variety of locations, the use of this technique has enabled a means of assessing which movement characteristics are most discriminatory, alongside producing accurate models.

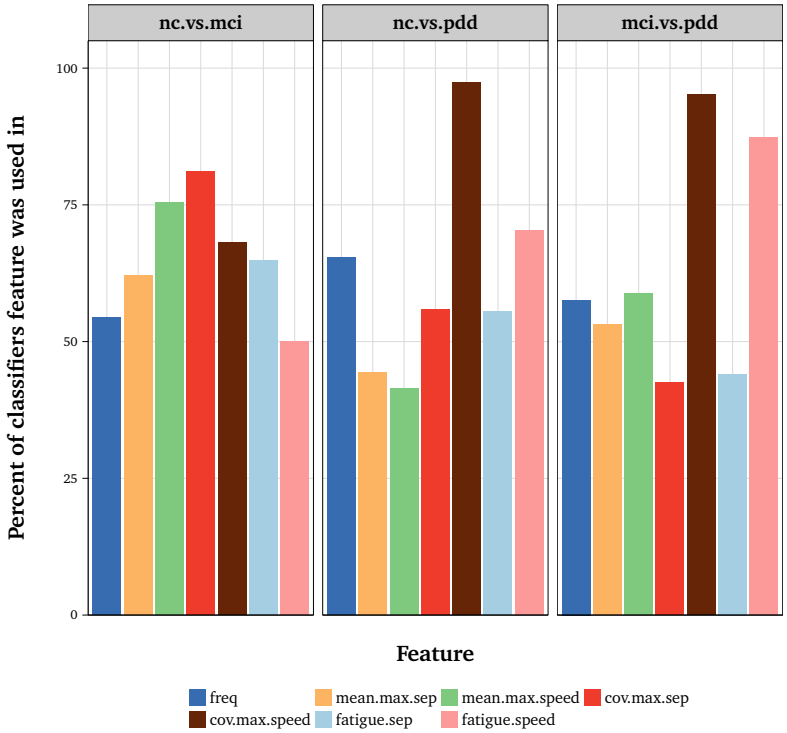


Figure 6.9: Bradykinesia feature usage rates for the cognitive impairment simulations

6.5.5 Discussion

The results of both the cognitive modeling and the examination of the feature selection, highlight the significant degradation that Parkinson's Disease with Dementia (PDD) has on

a person's motor skills, especially compared to Parkinson's Disease (PD) patients classified as suffering with either no, or mild cognitive impairment. It has been shown that there is a link between a patient's age and the likelihood of them experiencing cognitive decline, which could also explain the additional deterioration of motor skill. The significant impact of PDD upon a person's movement disorder is supported by the finding that ensemble classifiers achieved far greater success in differentiating between PDD and either of the other groups than in the *nc.vs.mci* comparison, which did not improve over a random guess. Furthermore, the selection of bradykinesia measures was most similar in the two data sets comprising PDD patients, and thereby the data sets in which the model was able to achieve better than random discriminatory ability.

Using ensembles of expression trees provides a means of implicit feature selection, allowing for a simple insight into which movement components are useful in separating the two classes. This method was used previously to investigate the differences in movement between healthy control subjects and PD patients, and has been successfully applied here to discriminating between various levels of cognitive impairment. The analysis highlighted some key findings, for example several attributes are shown to be crucial to identifying late stage PD—including the variability and progressive decrement of speed, representing akinesia and the sequence effect respectively—while not being as discriminatory between less cognitively impaired patients.

6.6 Conclusions

Modeling the movement disorder markers derived in a clinically motivated investigation with the previously developed evolved ensembles method resulted in accurate differentiation between PD patients and healthy controls, alongside emphasising the importance in consistent experimental methodologies when recording data at multiple locations. Combining data recorded under subtly different conditions allowed for the building of generalised models which achieved better AUC values than classifiers trained solely on data recorded at the independent centres.

In addition, a patient's cognitive level could also be modeled with success using the same summary movement features. The use of tree based base classifiers with their inherent feature selection allowed for a succinct insight into the specifics of the movement characteristics which identify PD, and also into the differing severities of the disease. The work described in this

chapter has emphasised considerations to make when modeling real world data, rather than freely available benchmark data sets which have already been processed and cleaned. The importance of consistent data recording strategies has been highlighted, in addition to being able to determine which features have been deemed by the evolutionary learning algorithm to be most indicative of Parkinsonian movement disorders. This enables feedback to be provided to the PD experts who may have their own preconceived notions of the most discriminatory movement components.

The next chapter builds upon this work to investigate using Evolutionary Algorithms (EAs) and other CI techniques to automatically extract summary measures from the raw movement data without requiring the expert clinical knowledge used in this chapter to form the seven markers.

Chapter 7

Modeling Time Series Data with Genetic Programming

Contents

7.1 Introduction

Fitting a classifier to data is an optimisation process to find a discrimination function that is most successful at mapping a given set of features onto a class prediction. The data attributes can take on numerous forms as both continuous or categorical descriptors, and typically have

been extracted from the raw data using either domain knowledge or a heuristic. The choice of which features to include in the classification model can have large repercussions on the final model's accuracy and as such *feature extraction* and *feature selection* strategies are well studied in the literature (Kira & Rendell, 1992; Kohavi & John, 1997; Guyon & Elisseeff, 2003). However, for problems with a temporal dimension, summarising the raw data into a discrete set of values can result in a loss of useful information and places greater importance on the feature design process. For diseases such as Parkinson's Disease (PD), it can be challenging to identify useful summary measures from high resolution time series measurements without expert guidance, and depending on the problem there can be a lack of domain knowledge. Typically, a procedure of trial-and-error is used when forming such classification models, rather than a systematic approach.

An automated form of feature design for time series data would thereby greatly aid progress towards a fully objective diagnosis of PD from simple movement data, without requiring the input of medical experts. Such a technique would also offer advantages for the clinical neurology research community, as it could provide an insight into discriminatory movement components highlighted by the feature construction process that may not have been considered previously. Evolving expression trees with Evolutionary Algorithms (EAs) provides a convenient solution to this problem, as expression trees are flexible in their representation of input data and functional ability, while EAs provide an objective search process which does not require any knowledge of the underlying solution space, merely necessitating a measure of goodness of fit—easily provided by standard classification evaluation metrics. As discussed in Section 4.4.4, Genetic Programming (GP) has been successfully used for feature design owing to these properties.

This chapter outlines the extension of the windowed approach developed by Lones, Smith, Alty, Lacy, Possin, Jamieson, Tyrrell et al. (2014) to form a broader assessment of its suitability to model PD movement data, and establish its effectiveness compared to forming predictions from movement disorder markers extracted with clinical guidance. In particular, rather than using the original arbitrary window length, the periodic finger tapping data is pre-processed into its natural constituent cycles for input into the expression trees; two additional aggregation functions are investigated to represent akinesia and the sequence effect, alongside using the mean; and taking inspiration from the research into ensemble classifiers, the outputs of multiple trees are combined into a new data set as a means of dimensionality reduction by creating new features from the raw time series without requiring domain knowledge. This

GP based method is subsequently compared to classifiers operating on the previously seen bradykinesia measures from Chapter 6. Finally, this technique is also applied to predicting levels of cognitive impairment in addition to distinguishing between PD patients and healthy controls.

Two approaches are discussed with relation to constructed features. The first considers the summary feature created by the expression tree as an individual classifier, effectively reducing the dimensionality of the temporal separation data (comprising approximately 1,800 samples from thirty seconds recording at 60Hz) into a single value. Upon application of a threshold it can provide binary predictions; its discriminatory power can therefore be evaluated using Receiver Operating Characteristics (ROC) as before. An additional approach combines the output of multiple such classifiers into a new data set, thereby using feature design to form a secondary data set, similarly to the method employed by the Evolved Vote Aggregators (EVAs).

Section 7.2 introduces the Temporal Expression Tree Classifier (TETC) approach used to extract summary measures from the time series data. In Sections 7.3 and 7.4, the extracted features are evaluated as individual classifiers on the PD diagnosis and cognitive impairment data sets used in Chapter 6, along with an analysis of the resultant expressions to determine how the predictions are made from the raw movement data. Multiple constructed features are then combined into a further data set to be used to differentiate between the PD patients and healthy controls, to be compared to using the seven bradykinesia summary measures. These simulations are described in Section 7.5.

7.2 Temporal Expression Tree Classifier

7.2.1 Overview

Summarising time series into a set of discrete features typically requires domain knowledge to process the data; an alternative approach is to use meta-heuristics to search for an adept feature construction algorithm without requiring any expert interaction. Expression trees evolved using EAs have been selected for this purpose, as their flexible nature allows them to accept input from many forms, not just in the standard approach of a discrete set of attributes; in addition the function set can be manipulated to provide a domain appropriate set of computational tools. Upon the use of a fitness function—in this case model Area Under Curve (AUC)—EAs can guide the search to useful areas of the solution space where discriminatory functions can be readily produced.

The finger tapping recordings consist of periodic movement, allowing for the creation of both global and local (to each tapping cycle) summary measures. Six of the seven measures used in the modeling of bradykinesia (as shown in Table 6.2) measure local aspects of each tap—the maximum separation and speed—before aggregating these values across all cycles to produce a global summary. The remaining measure—tap frequency—only summarises the recording at the global level. The approach taken by this objective feature design process is to form new local functions, outputting a continuous value for each tap cycle. As with the extraction of the bradykinesia markers then, a means of aggregating these tap attributes into a single summary measure per recording will also be required. The aim is to evolve expression trees which can identify discriminatory, non-linear measures in the separation profile from a finger tapping cycle. For ease of convenience, models built using this technique are termed Temporal Expression Tree Classifiers (TETCs) throughout this chapter and the remainder of the thesis.

7.2.2 Implementation

As detailed in Section 6.2.4, the raw positional data from the two Electro-Magnetic (EM) sensors is pre-processed to produce a smooth trace of separation between the index finger and thumb, which is then segmented into distinct tap cycles by locating local maxima in the waveform. As each tap sequence contains a varying number of separation samples, normalisation needs to be applied to the data points to obtain a consistent number of values to be input into the evolved expression. This was achieved by using an input set of ratio values in the range $[0, 1]$, representing the offset in the tap window from which to obtain the value of separation for each particular input node. For example, if an input tap sequence comprised the ten normalised separation values $[0, 0.2, 0.45, 0.7, 0.8, 0.85, 0.8, 0.6, 0.4, 0.2]$ (in practice a tap generally lasts just under a second, corresponding to slightly less than sixty samples per tap at the recording rate of 60Hz), a terminal input node with the ratio value of 0.4 would produce a value of 0.7 for that node's output. This procedure is shown in further detail in Figure 7.1. Features formed in this manner will thereby classify on the basis on the shape of the input separation waveform. The offset values are initialised randomly in the range $[0, 1]$ and are tuned throughout the evolutionary process by point mutation, allowing for an objective means of locating the most discriminatory portions of the finger tapping cycle.

To enable the derivation of non-linear functions mapping the input separation samples to a score, the expression trees will utilise an arithmetic function set, comprising the values $\{+, -, *, /, max, min\}$. To form an overall output for the recording, an aggregation function is

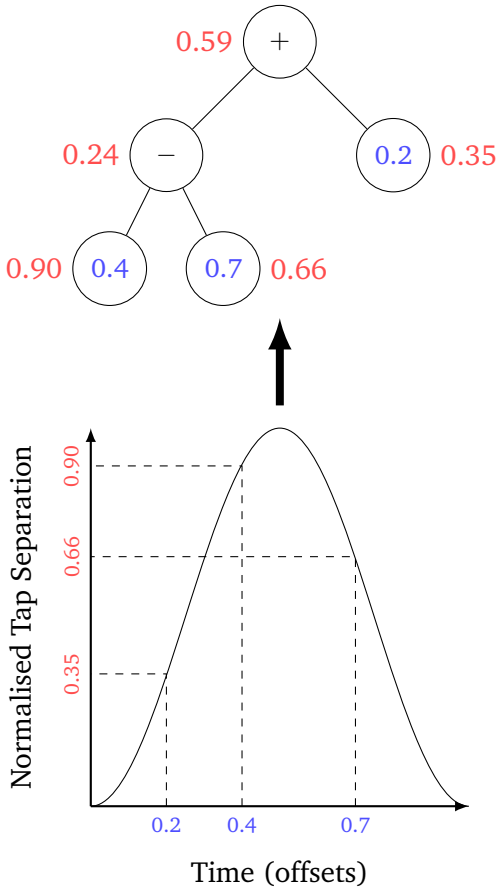


Figure 7.1: Input nodes' tap window ratios are converted into their corresponding separation value to produce an output score for each finger tapping cycle

required to summarise the output of each tap from the evolved expression. Taking inspiration from the derivation of the bradykinesia markers, the same three aggregation functions were used for this role, specifically the *mean*, the *coefficient of variation*, and the *slope* of a fitted regression line. The use of these composite measures allows for different global trends to be identified in the data, without requiring domain knowledge.

The same EA settings as used for the training of the discrete feature classifiers (observed in Chapters 5 and 6) were implemented, i.e., a population size of two hundred, a one hundred generation limit, and the Deterministic Crowding (DC) breeding operator, however, ensembles were not formed from the resultant models.

Table 7.1: Data processing methods used in the TETC simulations

Description	Identifier
Bradykinesia Features	*BK.features*
TETC, aggregated with cov	*tetc.cov*
TETC, aggregated with mean	*tetc.mean*
TETC, aggregated with gradient	*tetc.regress*

7.3 Diagnosing Parkinson's Disease with Temporal Expression Tree Classifiers

7.3.1 Comparison to Bradykinesia Features

Methodology

To establish whether the feature design TETC approach is competitive with modeling the features extracted with expert clinical knowledge in the preceding chapter, the TETC algorithm was employed to build classification models on the same PD movement data as used previously. In particular, three versions of the TETC algorithm were used to train models to differentiate between healthy controls and PD patients, differing in their choice of tap score aggregation function as being either the *mean*, the *coefficient of variation*, or the *slope* of a simple regression line. This approach was evaluated on a new resampling of the PD data into 5 x 10-fold cross-validation, as the TETCs are more computationally demanding to train than the ensembles modeling the bradykinesia markers owing to the far larger dimensionality of the data set. The labels used to refer to these processing techniques are summarised in Table 7.1. The TETC EA was run thirty times on each training set, providing a variety of constructed features for later use in a combined data set (this work is detailed in Section 7.5), however the results described here are concerned with using the output of the TETCs as individual classifiers. Ensembles modeling the extracted bradykinesia markers from the previous chapter were also fitted to the same data folds, enabling a matched comparison between the objective feature design approach and that using domain knowledge.

Results

Figure 7.2 details the results of the comparison of the three TETC approaches and the extracted bradykinesia measure classifier. The most immediate result is that on data recorded at two of the three centres, the TETCs are competitive with the *BK.features* models, but on the

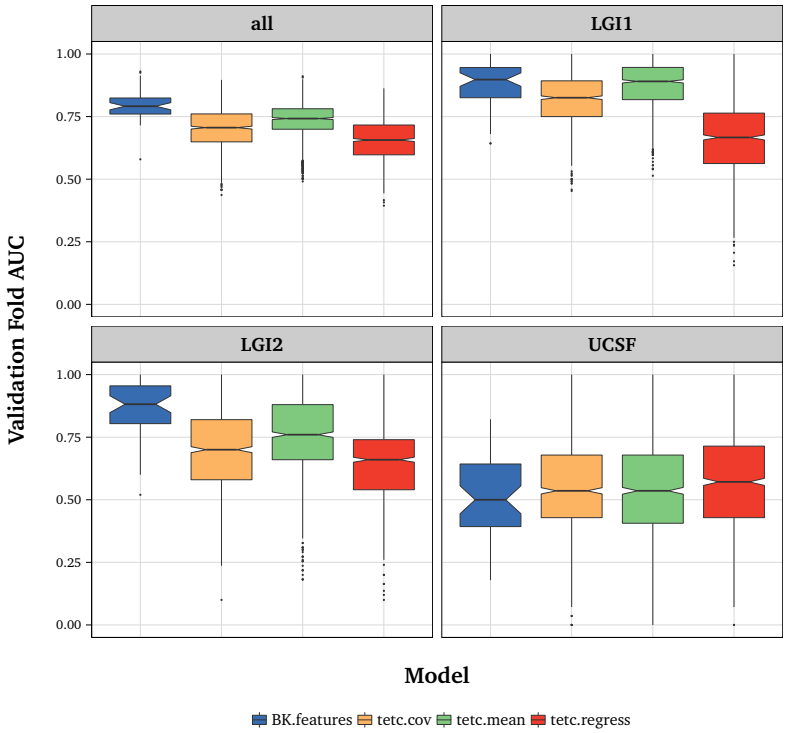


Figure 7.2: Comparison of the three TETC methods with the summary bradykinesia features

third—*LGI2*—the accuracy of classifiers modeling the raw separation data is significantly less than that obtained from the bradykinesia markers. This difference suggests that there is an aspect of the raw data recorded at this centre that presents challenges to the TETCs, but which does not pose a problem to the bradykinesia measures. As a result, on the data set comprising data recorded at all three centres, the *BK.features* attribute set has a significant improvement over the features constructed using the TETCs.

The choice of aggregation function is shown to significantly impact upon the model accuracy: the *mean* function is typically more accurate overall followed by *cov* and then the regression line slope. These latter two summary methods are attempting to identify global trends in the finger tapping data, similar to the bradykinesia attributes describing the amount of fatigue and hesitations. In contrast, the *mean* approach focuses on classifying data based on information local to each tap.

The Friedman test was statistically significant at the 0.1% level, indicating that the choice of data processing technique has a significant impact upon the overall model's accuracy. This was explored in further detail by a post-hoc Nemenyi multiple corrections test, which indicated that every pairwise combination of data processing methods is statistically significant at least

at the 5% level.

Discussion

The most striking aspect of the results is the difference in classification accuracy between the TETCs constructing their own summary features from the raw tap samples, and those using pre-determined measures on data recorded at one of the centres—*LGI2*. This serves to highlight the importance of effective data management techniques to ensure noise is reduced as much as possible. From inspection of Figure 7.2, it appears as if the extracted bradykinesia feature method was far superior to the TETCs, however on data recorded at the remaining two centres the latter were extremely competitive. In addition, the decreased performance of this technique on the *all* data set is presumably due to the inclusion of the data recorded at *LGI2* in the training samples. Further work will investigate the source of this noise and determine whether any steps need to be taken to prevent it from occurring in future recordings.

The choice of aggregation function for the TETC method appears to make a significant difference in the overall model accuracy, with the *tetc.mean* model being most effective overall, followed by *tetc.cov*. Each of these methods is attempting to find patterns in slightly different areas: the coefficient of variation function attempts to discriminate between healthy control subjects and PD patients by finding a tap measure which varies differently according to the subject type, the mean function is looking for measures which have an overall discriminatory nature between the two groups, while the regression line slope function attempts to form features based on decrementing (or incrementing) measures throughout the test duration.

When domain knowledge is available it can lead to simpler measures, for example *mean.max.sep* is a highly discriminatory feature and yet the attributes produced by the TETCs are likely to represent more complex non-linear patterns in the separation data. However, provided that the TETC is able to indicate the most discriminatory sections of the time series—as will be demonstrated in the following section—then this is not an issue and can help with providing useful information back to the clinical neurology research community.

Overall, the results indicate that where domain knowledge is not available, using a minimum of pre-processing to shape the raw data into a form suitable to be input into an evolved expression is competitive with extracting highly descriptive measures with expert knowledge. Indeed, the *tetc.mean* model achieved a median AUC on the *all* data set of 0.74, which is indicative of notable discriminatory ability. Furthermore, the TETC is more adaptable to new sources of data. For example, if it were decided that modeling an additional Unified

Parkinson's Disease Rating Scale (UPDRS) movement task would aid diagnostic ability, then the TETC method would be able to model the new data once it had undergone pre-processing into constituent periodic cycles. Extracting relevant summary features on the other hand would be more time consuming since it would require the guidance of an expert on PD. However, care must be taken when using the TETC technique as it appears that in some cases there is noise evident in the raw data which is filtered out by the extracted summary measures, but can lead to decreased performance in models relying upon the raw data.

7.3.2 Analysis of the Temporal Expression Tree Classifiers

Rather than using prior extracted features, the TETCs reduce the high dimensionality movement data into a single summary measure, which is then used to form predictions by means of ROC. Examination of the evolved expression can provide an insight into which components of the tapping motion are being used to distinguish between healthy subjects and PD patients. This can be simply achieved by observing the ratios of the tap cycle, which have been randomly selected by simulated evolution to produce more accurate models, thereby providing an objective means of observing the most discriminatory portions of the finger tapping cycle. Furthermore, since each finger tap is scored independently with a higher value representing a greater chance of belonging to a PD patient, the most and least discriminatory tap cycles can be found from across the whole data set to help understand the model.

The scores of every tap in the *all* data set from the most accurate individual TETC for each of the three aggregation methods trained on the *all* data set were determined, with the separation profile of the one hundred taps most indicative of healthy test subjects and PD patients being shown in Figure 7.3. The cycle offsets, passed into the expression tree as inputs, are shown as dashed green vertical lines. First of all, the plot highlights that the main difference between healthy taps and those as deemed as coming from PD sufferers is in the shape of the waveform. The most healthy taps consist of a smooth sinusoid with a clear minimum at either end and the moment of maximum separation occurring at approximately halfway through the movement. On the contrary, taps with no clear peaks, or peaks at different points in the tap cycle are deemed to be indicative of PD. In addition, the baseline value is on average much higher for the most 'abnormal' finger taps, highlighting that PD sufferers fail to bring their index finger and thumb as close together, or change the manner in which they bring their digits together throughout the task duration. There is a subtle difference in the waveform of the highest and lowest scoring tap waveforms according to each aggregation function, indicating that these

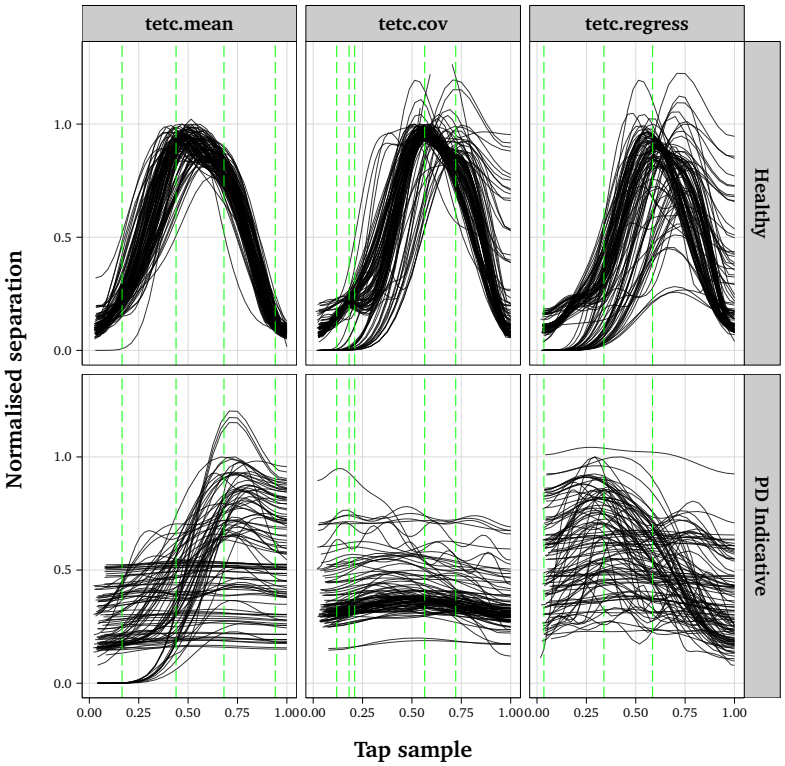


Figure 7.3: Visual comparison of the 100 most and least normal taps as determined by the three TETC methods

three methods are identifying diverse motifs in the raw movement data—a useful property for later combination into a composite model.

The window offsets used by the TETCs to form their predictions are typically located at the beginning, middle, and end of the cycle. Presumably then, a tap is identified as being PD indicative if it does not have a large difference between the starting separation value, the separation amplitude at approximately halfway through the cycle, and the ending; in other words a sinusoidal wave. However, there are again subtle differences in offset placement across the aggregation methods. For example, the offsets selected by the *tetc.mean* model are evenly distributed across the window length, with two offsets calculating the separation at either side of the peak. This has the effect of identifying tap waveforms as healthy if the peak is located at just after half the window length; some taps with the point of maximum separation occurring at a tap ratio of 0.75 are classified as being PD indicative despite having a relatively well defined sinusoidal shape. Identifying the discriminatory sections of the tapping cycle from the TETCs using the coefficient of variation and regression line slope proves to be more challenging, owing to the manner in which they identify global discriminatory patterns in addition to those local to each finger tapping cycle. It appears that the *tetc.cov* model is

forming a similar measure to the *tetc.mean* version, thereby classifying a recording based on how much this measure varies over the thirty seconds. The *tetc.regress* model on the other hand appears to generate a simpler summary feature by using only three offsets in the first half of the tap window, thus distinguishing between PD and healthy controls by how much an aspect of the finger opening phase degrades over time. This could indicate that the sequence effect is due to difficulties in extending joints, rather than contracting them.

The use of window offsets to parse the high dimensional data thereby allows for a detailed insight into how the predictions are being made. It also highlights that since the modeling relies upon accurate identification of tap cycles from the raw data, care must be taken in the pre-processing stage to segment these as accurately as possible.

7.4 Identifying Cognitive Impairment with Temporal Expression Tree Classifiers

7.4.1 Comparison to Bradykinesia Features

Methodology

TETCs were also trained to distinguish between the varying levels of cognitive impairment, which was previously investigated using the extracted bradykinesia markers as described in Section 6.5. To save computational expense, only the most accurate aggregation function—determined in the previous section to be the mean—was selected for these simulations. The models were trained on the same 30 x 10-fold cross-validation resamples as with the bradykinesia measure classifiers, enabling a thorough comparison between using domain knowledge to form discrete measures, and inputting raw data directly into the model.

Results

The results of the comparison between using the established bradykinesia measures and the TETCs modeling the raw data are shown in Figure 7.4. As with the movement data recorded from the various clinics, using the raw separation trace can produce competitive results to applying domain knowledge and extracting discrete features. However, as with the previous findings, on one data set (*nc.vs.pdd*) the classifiers using the raw tap traces were significantly less accurate than the extracted bradykinesia measures, but still providing a

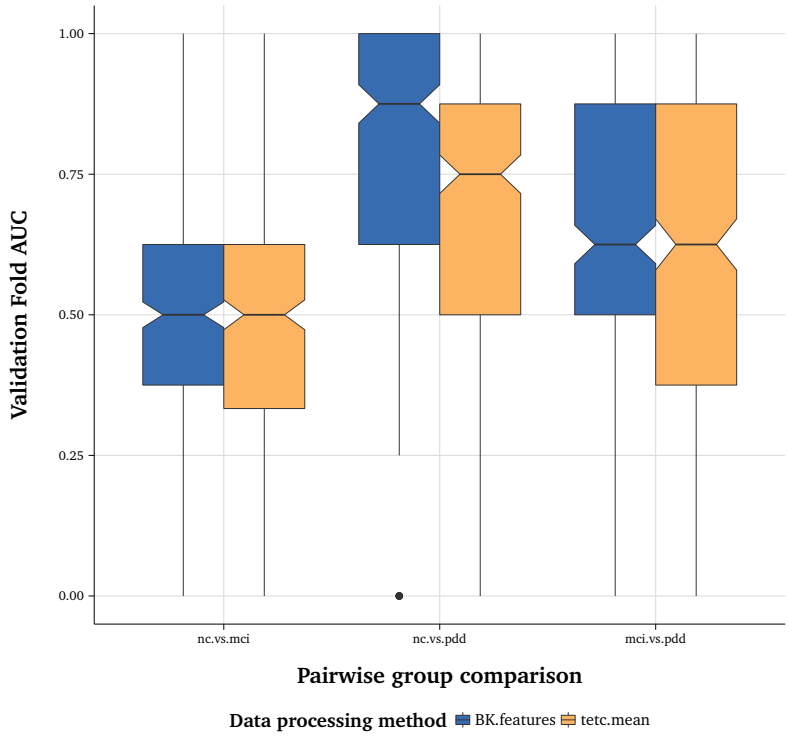


Figure 7.4: Comparison of TETCs and the summary bradykinesia features on the cognitive data sets

useful model (median AUC = 0.75). Again, the relatively large range of classifier accuracies is due to the choice of ten-fold cross-validation resampling strategy and the limited data set. On the *nc.vs.mci* data, the TETCs are still unable to improve upon a random guess, highlighting the amount of noise present in these samples hindering predictive power, either due to overtraining or intrinsic error. The Wilcoxon matched signed-rank test produced a p-value < 0.1, demonstrating that while the two methods of data processing result in similar AUC values on two of the data sets, overall the difference is statistically significant.

Discussion

The results of the comparison between using the previously formed movement disorder markers and passing raw movement data into the classifiers demonstrated that using domain knowledge to extract discrete features is generally the preferred form of data processing. Modeling the raw data with minimal processing can also produce accurate classifiers on average, however, as with the results from the PD identification runs, certain data sets appear to contain noise in some form, resulting in significant decreases in accuracy for the TETCs.

Further work will investigate the cause of this issue so that robust, objective, and accurate classifiers can be trained using a minimum of domain knowledge.

7.4.2 Analysis of the Temporal Expression Tree Classifiers

Previously, analysing the TETCs offered an insight into how PD is exhibited in a finger tapping task and compared this to healthy controls. By applying the same technique to the models fitted on the PD cognitive impairment data sets, subtle differences between the three levels of mental functionality can be shown. The one hundred most and least discriminatory taps according to the most accurate TETC for each data set are shown in Figure 7.5, highlighting several key findings. First, the offsets used by the classifiers modeling the cognitive data are relatively similar to those picked by evolution during the healthy control vs PD problem, whereby offsets are located at the maximum and minima of the signal (although the *nc.vs.mci* data set solely takes the peak separation into consideration). As before, the classifier is forming its predictions based on the shape of the finger tapping waveform.

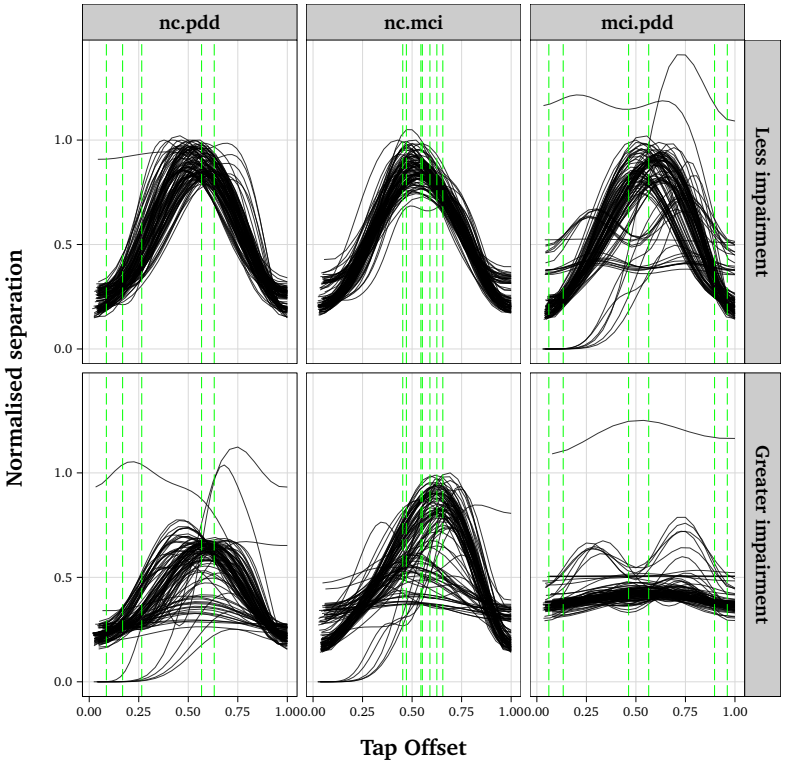


Figure 7.5: Comparison of healthy and abnormal taps in the cognitive data sets

In the two data sets containing PD patients showing no cognitive impairment, the most 'normal' taps look extremely similar to those from healthy control subjects comprising clean sinusoids,

however, in the *mci.vs.pdd* data set even the most healthy taps show signs of impairment. Likewise, in the *nc.vs.mci* data set, the most 'abnormal' taps are much more recognisable as sinusoids than from the other two data sets, highlighting the difficulty in discriminating between these closely separated groups. It appears that for this data set the prediction is formed from the time of the maximum separation, as there are several offsets around the 0.60 tap duration mark, corresponding to a peak in the most 'abnormal' data but at this time the healthiest taps have entered the downwards slope. This is a curious finding as it indicates that patients with Mild Cognitive Impairment (MCI) take a slightly longer time to reach full separation than those without any cognitive impairment, however, it must be remembered that the model accuracy on this data set was little more than a random guess, and so these findings may not be hold true for the wider PD population. Another interesting finding is the double peak found in the most 'abnormal' taps of the *mci.vs.pdd* data set, suggesting that the subject is suffering from hesitations, similar to akinesia.

The difference in tapping movements between the three levels of cognitive functionality is much more subtle than the difference between healthy controls and overall PD patients, as seen previously. Interestingly, the TETCs still use similar segments of the tapping cycle to form their predictions, generally considering the overall shape of the resulting finger tapping separation signal. Analysis of the models highlighted interesting findings such as the delayed separation of PD-MCI patients, and the dual peak motion of Parkinson's Disease with Dementia (PDD) sufferers.

7.5 Objective Data set Construction

7.5.1 Overview

TETCs reduce the high dimensionality time series movement data into a single—potentially non-linear—feature, which is then assessed for its discriminatory ability using ROC. This process can be considered as an objective form of *feature design*, as opposed to having medical experts guide a *feature extraction* approach to extract summary values. Rather than using these newly created features as individual classifiers, the output of multiple TETCs can be combined to form a new data set of objectively created measures, similarly to how the EVA ensemble voting method comprised a secondary classification stage. This section discusses forming new data sets of such features, in addition to supplementing the bradykinesia attributes in an attempt to improve classification accuracy.

Table 7.2: Data Processing Methods

Description	Identifier
Bradykinesia features	*BK.features*
Bradykinesia features + 3 TETC features	*BK.features.ext*
Twenty-one TETC features	*gp.constructed*
Individual TETC aggregated with mean	*tetc.mean*

7.5.2 Methodology

Two new feature sets summarising the finger tapping data recorded at the different sites were formed, the first consisting of the seven original bradykinesia features (Table 6.2) in addition to three more, comprising the output of the most accurate TETCs on the training set using the *mean*, *cov*, and the *regression line slope* aggregation methods. The second new data set required no domain knowledge, instead containing the features created by the seven most accurate TETCs as determined on the training set for each of the three summary functions, resulting in a feature vector with twenty-one continuous attributes.

The most accurate classification parameters as determined by the experiments on the University of California Irvine (UCI) data sets (Chapter 5) were used to model these two new objectively created data sets with GP arithmetic expression tree base classifiers evolved using DC and the *features* similarity measure, combined into ensembles of size twenty selected by elitism and having their votes aggregated by the evolved average approach.

To place the new results into context, they will be compared against the AUC scores from models trained on the bradykinesia markers, as well as the *tetc.mean* model. The four different data processing methods in this experiment are summarised in Table 7.2. The classifiers were assessed on all four of the PD diagnosis data sets, to establish whether combining objectively created features is still subject to the lack in performance on the *LGI2* samples, seen in the raw expression tree models.

7.5.3 Results

Table 7.3 details the overall mean AUC of classifiers formed using the four data processing techniques, while Figure 7.6 displays a plot of all the validation fold scores for each technique across the data set. The Friedman test produced a p-value < 0.1, indicating that the choice of data processing method has a significant impact upon the resultant model's AUC value. The results of the subsequent post-hoc analysis are displayed in Table 7.4.

Several key findings from the objective feature construction are immediately highlighted. First, extending the bradykinesia measures by adding features designed by TETCs does not aid classification ability, as evidenced by the reduced mean AUC and lack of statistical significance. However, it is interesting to note that the *BK.features.ext* models are the most accurate on the *all* data, which is the most important result when considering an application as this would incorporate all available data. Interestingly, on the *LGI2* data set, the addition of the constructed features actually causes a decrement in performance from just using the original markers, presumably resulting from the corresponding decrease in AUC of the individual TETCs.

The models trained on the constructed features data are not as effective as the bradykinesia markers, and present some unexpected results when compared to using the *tetc.mean* models individually. On the *LGI2* data set, where the TETCs struggled, forming a secondary feature vector with the *gp.constructed* technique is shown to be slightly beneficial, however this advantage is lost on the most separable data, coming from *LGI1*. Overall, the difference between constructing objective features and using the raw expressions is not shown to be statistically significant. This finding contrasts with the improvement offered by the EVA ensembles over their strongest member, which are formed in a similar manner to the *gp.constructed* models. None of these techniques, however, were able to produce classifiers moderately better than random guesses on the UCSF data set.

Table 7.3: Summary comparison of the data processing techniques

Data processing method	Mean AUC	SD
BK.features	0.763	0.187
BK.features.ext	0.754	0.199
gp.constructed	0.727	0.188
tetc.mean	0.722	0.185

Table 7.4: Post hoc pairwise p-values between data processing methods

	BK.features	*BK.features.ext*	*gp.constructed*
BK.features.ext	0.99		
gp.constructed	$p < 0.1\%$	$p < 0.1\%$	
tetc.mean	$p < 0.1\%$	$p < 0.1\%$	0.45

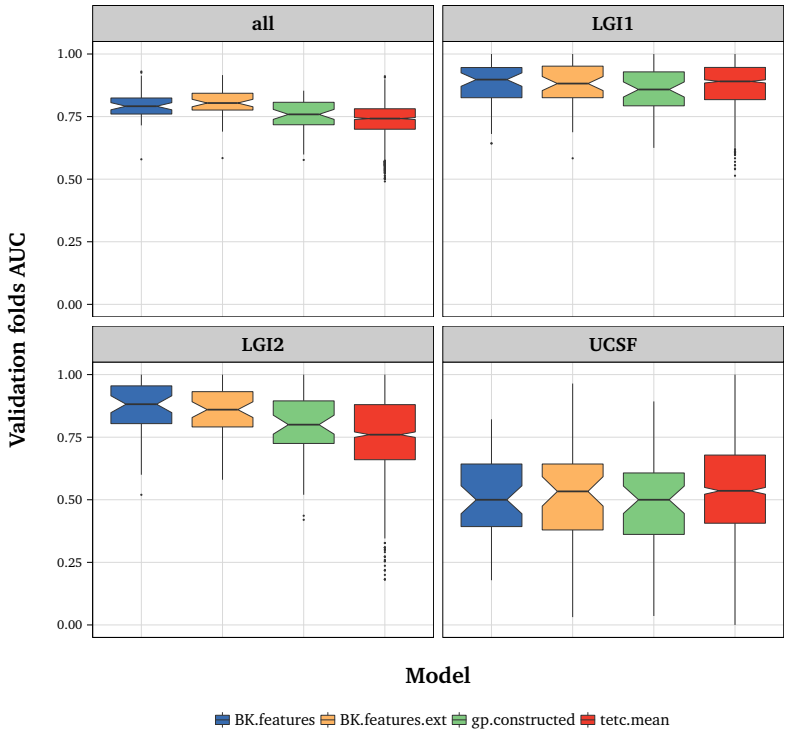


Figure 7.6: Comparison of the constructed feature set approaches with the previous models

7.5.4 Discussion

Analysing periodic data neatly allows for the raw data to be segmented into constituent cycles, from which a number of aggregation functions can be used to produce a single output score for each data pattern, enabling a variety of movement dynamics to be modeled. Combining the outputs of multiple such feature design algorithms facilitates the generation of a secondary data set. While a number of different expression trees would need to be evolved to form a competitive data set (in this instance twenty-one were used), this process would fit in neatly with standard GP practice whereby several runs are typically performed at the same time to obtain an estimate of variance. While forming this objectively created feature set did not improve upon the accuracy of models trained on markers extracted using domain knowledge, there is scope to further develop this approach and it is expected that a benefit can be gained in a similar fashion to the increase in AUC resulting from the secondary modeling stage of the EVAs. Furthermore, supplementing the measures derived with clinical guidance with objectively created features did not result in an increase in accuracy.

Therefore, for applied problems where optimal predictive power is required, using domain knowledge to reduce the input data dimensionality is the preferred option. Forming a

secondary data set from multiple TETCs' outputs can still produce models with strong discriminatory ability, but these do not significantly improve over using the individual TETCs as classifiers.

7.5.5 Visualising Feature Importance

Since the base classifiers used to model the secondary constructed feature data set were GP expression trees, the importance of each attribute can be simply visualised in the same manner as used in Chapter 6, by counting the usage frequency of each feature across the members of the ensemble. Specifically, this was counted across the twenty ensemble members formed on each of the five repeats of the ten data folds. The constructed data set comprised twenty-one features, corresponding to seven features from TETCs using each of the three aggregation functions.

The resultant constructed feature usage rates are displayed in Figure 7.7, grouped by aggregation function, and then within each of these groups by their rank when used as an individual classifier. On the *all*, *LGI2*, and *UCSF* data sets, there is a relatively balanced usage of features constructed using the three aggregation methods, contrasting with the significant difference in accuracies of each of these methods when used as individual classifiers (Figure 7.2). This suggests that by combining features summarising different components of movement, a more generalised prediction can be formed, similarly to how ensembles combine knowledge to be more accurate than the sum of its members. Models fitted to the *LGI1* data, however, disproportionately selected the *tetc.mean* features. Unsurprisingly, the most individually accurate features were more likely to be selected than the weaker ones.

7.6 Conclusions

The combination of EAs and expression trees has provided an objective means of reducing the dimensionality of movement data with no expert knowledge of the problem area required. Both of these fields are ideally suited for this technique, as expression trees provide a flexible model which can receive data input in numerous representations, and EAs are extremely useful for a situation such as this when no domain knowledge is available, and the parameters of an ideal summary feature are unknown. By exploiting both of these attributes, models can be formed using minimal pre-processing and only standard arithmetic operations, which are competitive to those using expert knowledge to calculate summary markers. By implementing

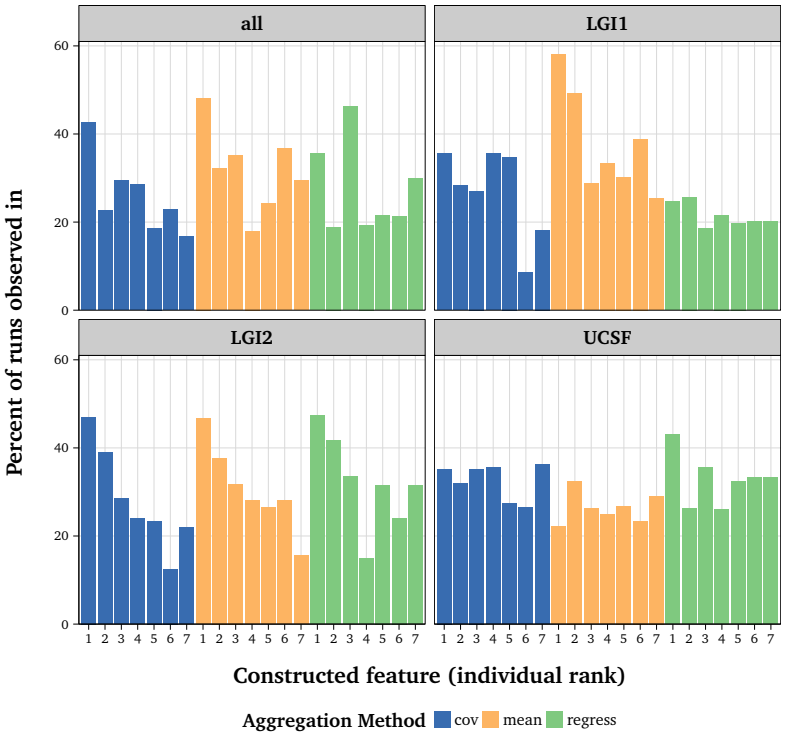


Figure 7.7: Usage rates of the constructed features from the most accurate models of the *gp.constructed* data set

an input set comprising window offsets, the objectively created features considered the shape of the separation waveform for each tap in its score calculation. Combining several such classifiers into a constructed feature set, similarly to the previous work on ensembles, was shown to not offer any significant advantages over using the TETCs themselves, however.

Both using the TETC output individually and forming a secondary data set was not shown to offer any benefits over modeling measures extracted with clinical guidance. However, the AUC scores of objective TETC classifiers indicate an effective amount of discriminatory ability to be used in a real world application, and so could be employed when domain knowledge is not available and still provide benefit. The use of an EA to evolve optimal window offsets facilitates an insight into which components of the finger tapping period are most discriminatory, while segmenting the data into constituent movement cycles enables a simple visualisation of the most and least discriminatory finger taps as determined by the TETC.

While the TETC approach has been able to successfully model the movement data, they have a small number of limitations which hinder their suitability to being used in a real world application. First, they are reliant upon an additional pre-processing stage to segment the

complete time series into its constituent finger tapping cycles. Therefore, the TETC would not necessarily be directly applicable to movement data from a new source, for example an additional UPDRS motor examination task. If this new source of data were periodic then a simple method would be to separate the raw sequence into its constituent cycles, however, this would still require time to develop a new pre-processing step. On the other hand, if a non-periodic data source was required to be modeled then the TETC method would fail to adapt, except if the time series were able to be subset into arbitrary segments, perhaps by application of a window. However, this would require experimentation to determine an appropriate window length.

Second, the TETC approach is only able to model the temporal dimension by use of the window offsets which are input to the tree; it offers no inherent method of modeling long-term dynamics or retaining a memory of previous tap cycles to provide a more accurate prediction. Finally, the TETCs are relatively slow to train, since each time series needs to be passed through every candidate solution in the population at each generation. The subsequent chapter describes an attempt to address these concerns by using another Computational Intelligence (CI) technique which is more inherently suited for modeling temporal dynamics and can be easily adapted to new sources of data without requiring further pre-processing steps, in addition to being significantly faster to train.

Chapter 8

Modeling Dynamic Movement Patterns with Echo State Networks

Contents

8.1 Introduction

The previous chapter widened the search for an automated Parkinson's Disease (PD) diagnostic aid to include a form of feature design using Genetic Programming (GP) operating on raw positional time series data, rather than summary measures extracted with domain knowledge. Forming new data sets from these constructed features was shown to produce competitive results for occasions when expert guidance is not available. However, this technique was not without its limitations for use as a practical diagnostic aid. In particular: human intervention would be required to provide a further pre-processing stage if a new data source were required to be modeled, since Temporal Expression Tree Classifiers (TETCs) are designed to operate

on periodic time series; TETCs are slow to train due to the population based search method and the high dimensionality of time series data; and expression trees offer no natural method of incorporating a memory of previous taps to inform their classification, since each cycle is evaluated independently.

This chapter extends the feature design approach by implementing a dynamical system to more accurately capture Parkinsonian motor disorder characteristics, in addition to requiring less pre-processing and being quicker to train than an Evolutionary Algorithm (EA). The specific choice of dynamical system is the Echo State Network (ESN), introduced in Section 4.3.5. Several considerations when adapting ESNs for classifying temporal data are discussed, including the processing of the raw data as well as model hyper-parameter choices. The accuracy of ESN classifiers are compared to the TETC approach, in addition to classifiers modeling the bradykinesia markers.

8.2 Motivation

As discussed in Section 7.6, an alternative method to TETCs for modeling movement data for use in an applied diagnostic aid was sought. Dynamical systems address two of the limitations of TETCs, particularly the concerns that TETCs require the data to be pre-processed into segments and that they score each constituent cycle independently as they do not contain a memory of previous inputs. Dynamical models inherently have short term memory of previous states, and can accept the input as a continuous stream rather than segmented into subsections. The most common dynamical systems in the Computational Intelligence (CI) literature are those which extend Multi-Layer Perceptrons (MLPs) with *context* layers keeping a reference to previous hidden activation levels. However, as detailed in Section 4.3.4, the learning algorithm for these methods is often complex as it involves unrolling the network at each timestep to calculate the error gradients for each weight. Reservoir computing offers an attractive alternative to Recurrent Neural Networks (RNNs) as they only require the linear readout weights from the reservoir to the output nodes to be trained, which is typically achieved with fitting a regularised linear regression—a method far less computationally expensive than backpropagation-through-time, or indeed the EA which fits TETCs. Out of the two most common reservoir computing algorithms—ESNs and Liquid State Machine (LSM)—ESNs offer the method which is easiest to use and is thus attractive for applied settings. As highlighted in Section 4.4.4, ESNs have not been commonly applied to classification problems,

and so this chapter also discusses the considerations to be made when using them in this scenario, particularly when used to model periodic time series.

To establish the suitability of ESNs to model the finger tapping data for classification purposes, a preliminary investigation was carried out to forecast a tapping separation signal. The network setup followed the procedure documented in Jaeger (2002) to establish suitable hyper-parameter values, which will not be described in further detail as it is beyond the scope of this thesis. The ESN was fitted on the first 70% of the signal, and then attempted to predict the remaining samples. The results are shown in Figure 8.1, where the vertical grey dashed line indicates the training/test partition, the predicted samples are coloured in red, and the actual signal is shown in black. While there is a constant bias error in the amplitude of the predicted signal, the network has captured the frequency of the finger tapping motion well, with an Root Mean Square Error (RMSE) of 0.266. These results highlight the ability of an ESN to accurately capture the finger tapping dynamics, and so this approach seems suitable for developing a classification model. The following sections describe implementing ESNs for this application in detail.

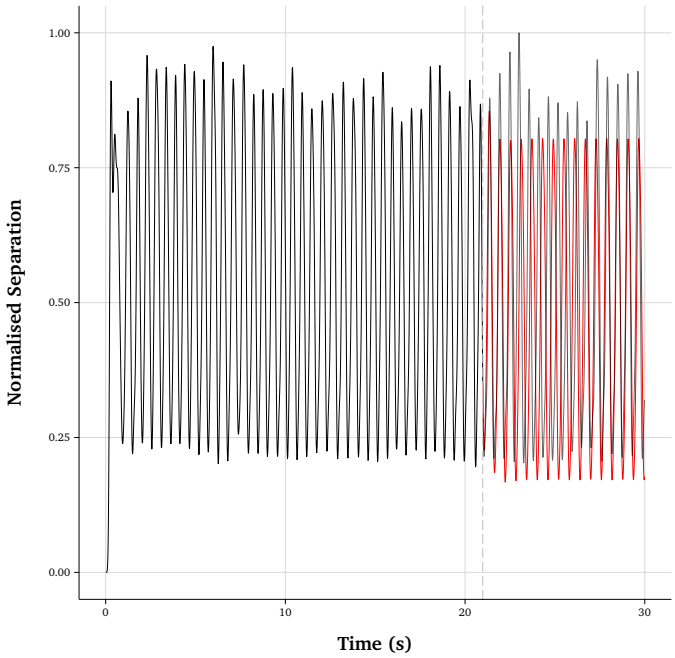


Figure 8.1: Forecasting a finger tapping signal using an ESN

8.3 Configuring Echo State Networks for Classification

8.3.1 Echo State Network Overview

As the field of ESN research is relatively recent and still expanding, an overview of practical implementation details are provided here for clarity.

Network Structure

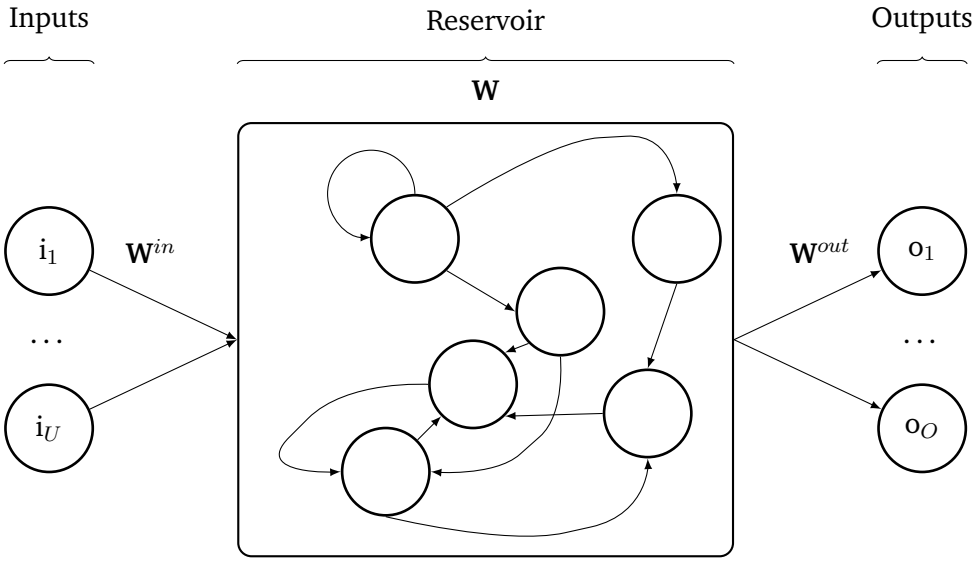


Figure 8.2: The reservoir of an ESN is sparsely connected with recurrent connections providing a means of maintaining state

The diagram of an ESN's structure that was originally shown in Figure 4.10 has been reproduced in Figure 8.2 for convenience. It highlights the fact that an ESN comprises three distinct sections: input nodes, a reservoir of computational neurons, and output nodes; a network's structure is thereby governed by three quantities:

1. The number of inputs at each time step - U
2. The size of the reservoir - N
3. The number of outputs - O

Of these values, U and O are dependent on the data being modeled, while N is a hyper-parameter to be optimised for each application for maximum efficacy. The connections between the inputs, the reservoir, and the outputs, can thus be described by the following three weight matrices:

- $\boldsymbol{W^{in}}$ - An $N \times U$ matrix containing the weights relating the network inputs to the reservoir
- $\boldsymbol{W}$ - An $N \times N$ matrix storing the weights of the intra-reservoir connections, allowing recurrent connections between nodes.
- $\boldsymbol{W^{out}}$ - An $O \times (U + N)$ matrix of weights providing an output from the reservoir and current input to the output

Initialisation

Optimising the reservoir dynamics is the primary focus of training ESNs, and so vastly more time is spent on optimising W than the other weight matrices. W^{in} and W are typically initialised randomly at each run, often with a uniform distribution of weights in the range $[0, 1)$; however, it has been shown that the specific choice of weight distribution for these matrices does not impact greatly upon network performance (Lukoševičius, 2012). For optimal network functionality however, the initialisation of W requires the tuning of two hyper-parameters—*sparsity* and the *spectral radius*. It has been demonstrated that ESNs function optimally with a *sparsely* connected reservoir, whereby a typical W would comprise only 1% of connection weights as non-zero. The value of sparsity determines the ratio of non-zero weights in W, and can also be interpreted in an absolute way, by dictating the minimum number of additional nodes that each node must be connected to regardless of the overall reservoir size.

The choice of value for spectral radius has been observed to play a greater role in a network's behaviour than its sparsity. The spectral radius of a reservoir is the maximal absolute eigenvalue of W; Jaeger (2001) demonstrated that ESNs are most effective when fulfilling the *echo state property*, meaning the spectral radius is less than one. A typical ESN initialisation procedure will scale W to provide a maximal absolute eigenvalue equal to the selected spectral radius value, not only to meet the echo state property, but also to tweak the network dynamics. The work described in this thesis using ESNs involved selecting values for sparsity and spectral radius for final model building by cross-validation.

The final connection weight matrix, W^{out}, is initialised to all zeros, as it is not used when running the time series through the network. After the network has run to produce an output for each time series, W^{out} is assigned weights to minimise the error between the output and the desired target signal. This forms the training procedure of ESNs and is described in greater

detail later on.

Running the Network

Once a network has been initialised with the two weight matrices described above, it can be evaluated over a time series. The behaviour of ESNs can be entirely characterised by two update equations, reproduced here in Equations 8.1 and 8.2. For an input time series $u(n)$, with S timesteps and the current timestep denoted by n, Equation 8.1 defines the update of the reservoir activation states ($x(n)$ with length N) performed at each timestep. An activation function $f(x)$ is required, this is typically a sigmoidal function such as $tanh$. This equation can be augmented with the use of leaky-integrated neurons, adding an additional hyper-parameter to the learning algorithm, however, the work described in this thesis does not use this approach. In addition, a bias term can be prepended to $u(n)$—acting in a similar fashion to the intercept term in a linear model—to introduce a greater range of dynamic behaviour. Finally, feedback connections from the output can be added at this stage, necessitating the use of an additional weight matrix W^{fb}, however, this is not used in classification problems.

$$x(n) = f(W^{in}u(n) + Wx(n-1)) \tag{8.1}$$

Once the activation states have been calculated at each timestep, they are stored along with the inputs in a $U + N$ vector, called the *extended states*, i.e. $e(n) = [u(n); x(n)]$. The network output can then be calculated using Equation 8.2, providing a linear readout from the reservoir. If desired, an additional activation function can be applied at this step, or the identity function can be used. It is important to note that in practice the extended state vectors for a predetermined number of initial timesteps are discarded to allow the network dynamics to dampen to a steady state, so that the reservoir activations are solely produced as a result of the input values and any feedback. The numeric constant dictating the number of timesteps to discard is termed T_0 and is typically determined by visual inspection of the internal reservoir states.

$$y(n) = W^{out}e(n) \tag{8.2}$$

Network Training

The training method for ESNs is unlike other forms of RNNs in that it only modifies the output weights W^{out}; once initialised, W^{in} and W remain constant. This is achieved by forming a linear regression equation, a general example of which is shown in Equation 8.3 where P represents the number of predictors in the data set, β represents the coefficients to be solved for, Y is the model output, and X are the data attributes. Note that the intercept β_0 is not included in this expression as it is not often incorporated into ESN training, although it can be added by appending a bias term to the inputs.

$$Y = \beta_1 X_1 + \beta_2 X_2 + ... + \beta_P X_P \tag{8.3}$$

This equation can be applied to ESN training by relating the output Y to $y(n)$ from Equation 8.2, where it is calculated as the sum of products between corresponding values in W^{out} and $e(n)$. The goal of the training method is to reduce the error term between the actual output and that from a desired target signal $d(n)$ (Equation 8.4). This is achieved by vertically stacking the outputs at each timestep into an $S \times O$ matrix Y, D is formed with the same dimensions as Y containing the target outputs, and the extended states are also stacked to produce an $S \times (U + N)$ matrix E. The weight update equation calculates the values of W^{out} to minimise the sum of squared residuals of this linear equation, as shown in Equation 8.5. Solving this expression using the same ordinary least squares method as for linear regression (with simple matrix transformations but this is beyond the scope of this thesis) will produce a matrix with dimensions $O \times (U + N)$, ready to be used as W^{out} for future prediction.

$$d(n) - y(n) \tag{8.4}$$

$$\sum_{i=1}^{S} (Y_i - E_i^T W_i^{out})^2 \tag{8.5}$$

An alternative to using linear regression to solve Equation 8.5 is to employ ridge regression, which extends the regression expression with a scalar penalty term λ, having the effect of reducing the absolute value of coefficients and thereby helping to prevent overfitting to the training samples. Equation 8.6 shows the derivation of the error term to be minimised when training an ESN with ridge regression. Ridge regression is a popular method of training ESNs despite requiring the tuning of an additional hyper-parameter (λ).

$$\sum_{i=1}^{S}(Y_i - E_i^T W_i^{out})^2 + \lambda \sum_{j=1}^{P} W_{ij}^{out} \tag{8.6}$$

8.3.2 Methodology

There are numerous considerations to be made when implementing ESNs to classify time series data, relating to both the network hyper-parameters and the amount of data processing to perform. Potentially the most significant choice is how to produce a single output prediction from the network being subjected to a sequence of samples. At every timestep the network is evaluated, providing each neuron in the reservoir with an activation level comprising the sum of its weighted inputs passed through an activation function. The reservoir output is then read as a linear readout from these values, resulting in as many network outputs—and thus possible classification predictions—as there are timesteps in the input data.

Referring back to the explanation of the ESN training method in Section 8.3.1, to calculate values of W^{out} to minimise the Mean Square Error (MSE) between $y(n)$ and $d(n)$, an $S \times (U + N)$ matrix of extended states is required, termed E, where S represents the number of samples in the training set. For forecasting applications S is simply equal to the number of timesteps in the training signal, however for classification problems where there are S independent training series, each with potentially varying number of timesteps, a means of providing a single $(U + N)$ extended state vector for each pattern is required. Lukoševičius (2012) recommends determining the overall output as the mean of the outputs at each point in time. In practice this involves calculating $e(n)$ for each timestep n, and deriving the mean values to produce a single vector of activation levels per data pattern. Appending these mean values to E results in an $S \times (U + N)$ matrix, as required to use Equation 8.5 or 8.6 to determine values for W^{out}.

An alternative means of providing a single vector of extended state activation values for each time series, is to use the values of $e(n)$ present during the final timestep; owing to the recursive connections this will be a function of all the previous state values. This is the approach followed by Verplancke et al. (2010). To better understand the characteristics of ESNs and their application to classifying movement data, this experiment will investigate both of these methods.

The next issue to consider is how to process the data for input into the network. Previously,

the TETCs received the input data tap by tap for two reasons: it provided a convenient method of reducing the dimensionality of the raw data, and since the expression tree had no means of remembering past values, passing the data in sample by sample would be meaningless. ESNs however, contain memory, and so passing the samples in point by point as was the case when forecasting the finger tapping is a possible data input method. Inputting the data in finger tapping windows, in the same way as with the TETC, is an alternative option.

Since the number of inputs to an ESN is fixed, unlike expression trees evolved with GP, a decision needs to be made on how to parse the raw data if it is to be input tap by tap. The approach taken for this work was to provide twenty input neurons, sampling the separation data from each tap at linearly separated offsets. Unlike the TETC approach, this does not allow the model to fine tune the specific offset values, but twenty samples allows for the capturing of the overall characteristics of the tap waveform shape. Both of these methods, passing in the data sample by sample, or tap by tap, were investigated during this work. As a reminder, the finger tapping separation data was initially sampled at a frequency of 60Hz over thirty seconds, providing 1,800 data points per data instance and the average number of taps in a recording was 82.5.

As discussed in Section 8.3.1, the standard training algorithm for ESNs solely modifies the output weights; the reservoir and input weights are initialised randomly. A brief preliminary investigation into using EAs to optimise the reservoir weights was conducted, but found that despite the considerable additional computational expense, network accuracy did not improve significantly. Further work could investigate whether this procedure could be improved, as it has been used successfully by Chatzidimitriou & Mitkas (2010) and Jiang et al. (2008), but that is beyond the scope of this research. The weight fitting method used in this work was ridge regression, whereby the value of the lambda penalty is determined by cross-validation. Using ridge regression helps to regularise the network to combat the overfitting seen when calculating the output weights with standard linear regression. Cross-validation was used to select optimal values for the hyper-parameters reservoir size, sparsity, spectral radius, and λ. To reduce the effects of the initial transient behaviour of the reservoir, values of T_0 for the single sample case and the tap-by-tap input method were selected as fifty and five respectively, determined by visual inspection of the network activation levels.

Table 8.1 summarises the different aspects of modeling the finger tapping data using ESNs along with the labels used to refer to these factors for the remainder of the chapter. The networks were trained on the positional data recorded at the three different international sites,

Table 8.1: Facets of ESN classification under investigation

Factor	Possible Values	
Data input type	Single sample (*single*)	Tap by tap (*tap*)
State aggregation method	Mean (*mean*)	Last state (*last*)

along with a combined data set. The same cross-validation resampling splits as implemented previously for the PD diagnosis data sets were used, allowing for a direct comparison with the bradykinesia marker data set and features created with the TETC method.

8.3.3 Results

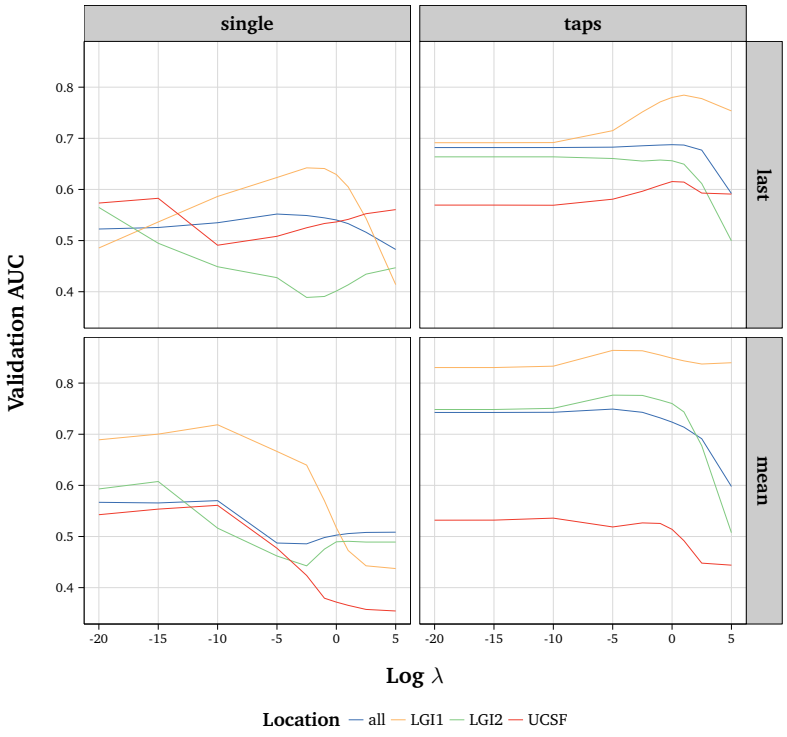


Figure 8.3: Tuning the λ regularisation parameter

To facilitate maximum generalisation ability, the λ regularisation hyper-parameter—used to fit the reservoir output weights—was swept across a range of values in a logarithmic grid. The results of this search are detailed in Figure 8.3, stratified by data processing method and state aggregation function, with each line colour representing a different medical centre. The impact of this penalty upon the generalisation ability of the network is shown to be significant, and varies according to the method of inputting the data into the network. For the remainder

of this section, the optimal λ values are used for each permutation of results. As ESNs have not been widely used for classification tasks before, there is not a set of well refined design guidelines to follow. As a result, this work investigates the impact of configuration choices on final model accuracy to better understand how to optimise ESNs for classification problems. Two of the most signification considerations are how to format the data for entry into the network, and how to establish a single one-dimensional vector of state values for each time series pattern. As highlighted in Table 8.1, two possibilities were investigated for each of these choices: *mean* and *last* state aggregation functions and *single* and *taps* data processing method.

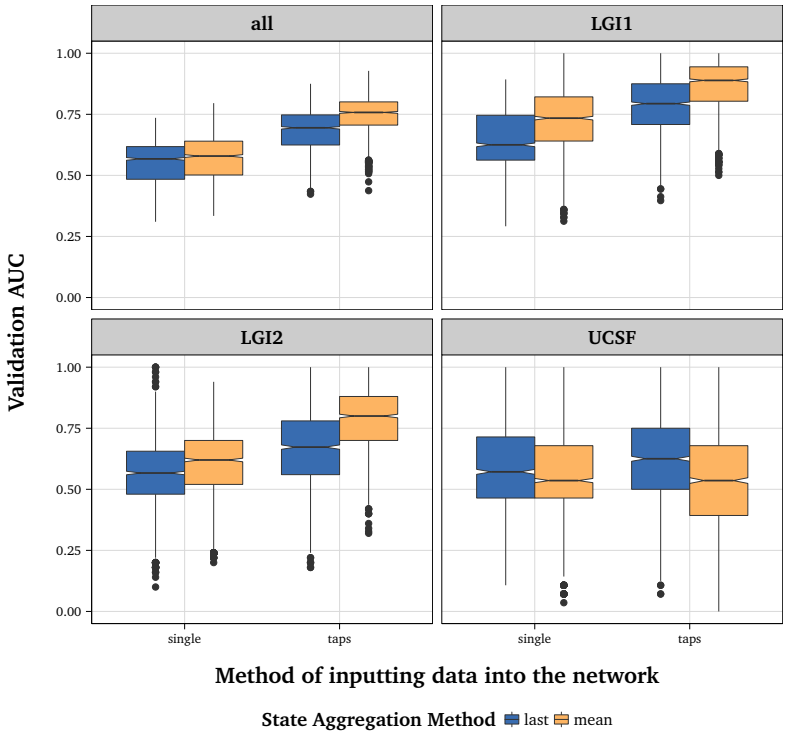


Figure 8.4: Comparison of ESN configurations for classifying periodic time series

Figure 8.4 displays the impact of the choice of state summary functions and data input procedures on the model Area Under Curve (AUC), as measured on the validation fold providing an unbiased view of the classifier's generalising capabilities. Immediately from the plot, the impact of each of these decisions can be seen. On every data set besides *UCSF*—which has demonstrated poor performance prior—calculating the overall state of the network as the *mean* of the states at each timestep is shown to be significantly advantageous over using the final state value, this result is supported by a p-value < 0.1 (determined by the Wilcoxon

signed-rank test). This finding is interesting as the final reservoir state is a function of every previous value, and so could be expected to provide a thorough prediction taking into account the entire recording, while the *mean* method incorporates reservoir neuron outputs produced early on in the time series when it could be assumedly harder to classify as less information about the recording is available. In addition, the *mean* function does not take into account the ordering of the tap inputs directly, except that each state is a function of its previous value.

The second issue under investigation was the method by which the data is presented to the network. Two different approaches have been seen previously—discrete summary attributes and offsets into a tapping window—however, as ESNs comprise memory of previous states, it is possible to pass the data in sample by sample to allow the network an insight into the global dynamics rather than focusing on the intra-tap aspects of movement. The results indicate that the networks achieve greater discriminatory ability by identifying trends within tap cycles, similar to how the TETC functioned. Hypothesis testing showed this difference to be statistically significant at the 0.1% level. While this approach is only suitable for periodic movements, it is a useful finding, particularly as less computational time is required to train networks due to the lowered number of timesteps. Interestingly, the experimental setup which takes most advantage of the temporal nature of the data is the *single last* combination, however, this configuration produces the least accurate models overall, indicating that the ESNs are not exploiting the recurrent connections to analyse the time dimension.

8.4 Constructing Feature Sets

8.4.1 Methodology

As with the TETCs, these networks achieve feature design by reducing the time series data into a single continuous value, which, when analysed with Receiver Operating Characteristics (ROC), can provide a measure of predictive power. An alternative option is to combine the output from numerous models into a new, objectively created feature set to be used in a secondary classification process. To facilitate this, thirty networks were created for each training fold under different seeds to provide a range of initial conditions, to produce a variety of predictive models to form the secondary data set from. This objective feature set technique was previously shown to not offer any advantages over using TETCs individually, however, there is scope for it to aid ESN classification, as the variety of configuration choices can offer a greater range of network dynamics to be modeled.

From this collection of ESNs, two different constructed feature sets were formed for each fold of the cross-validation iteration. Since the *taps* data input method combined with the *mean* state aggregation function was the most accurate configuration for the networks individually, the first constructed feature set comprised solely the outputs from the ten most accurate networks from these parameter settings as observed on the training set. This approach is labelled *meantaps*. To obtain a variety of features analysing different components of the finger tapping movements, the second feature design method incorporated outputs from each of the four permutations of data processing method and state aggregation function. The outputs of the three most accurate networks built with each of these parameter combinations were collected into a data set known as *combined*, comprising twelve total attributes. Referring back to the *exploration/exploitation* trade-off first discussed in the context of EAs searches, the *meantaps* method takes more of an exploitative approach by focusing on the most accurate classification method found by the raw classifiers. The *combined* data set, however, has more in common with the diverse searches used for ensemble generation, hoping to produce a more robust prediction by combining classifiers which have found contrasting discriminatory patterns in the raw data. Once formed, these objectively created data sets were used to model PD using the same ensemble setup as discussed in Chapter 7, providing for a fair comparison between the two movement data classification techniques.

8.4.2 Results

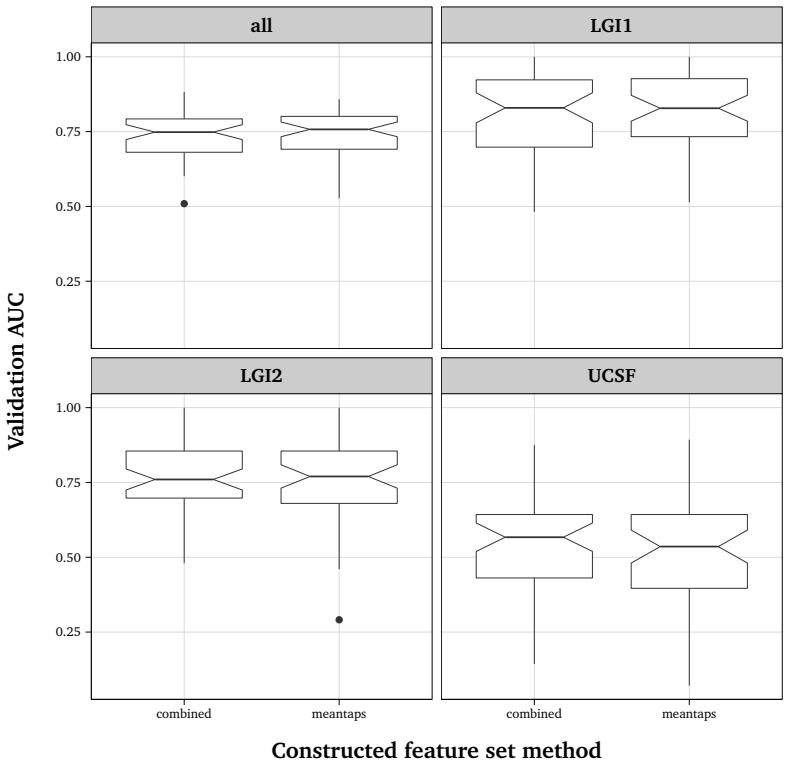


Figure 8.5: Comparison of feature construction approaches

The two feature set methods are compared in Figure 8.5 across each finger tapping recording location for their diagnostic efficacy. Perhaps surprisingly there is not a significant difference between the two approaches that can be observed visually from the plot, which is supported by the hypothesis test providing a p-value of 0.637. A possible explanation for this behaviour is that the use of evolved expression trees to model the constructed feature data set provides inherent feature selection, allowing for the weaker constructed attributes, such as those built from networks analysing the *single* sample data, to be unused by the final model. The mean validation fold AUC for each method was 0.708 for the *tapmeans* and that of the *combined* technique is 0.714, highlighting a slight bias towards the latter method.

To investigate whether the process of combining the output from multiple networks into a secondary data set offers any improvements over using the ESNs individually, a comparison was made between the most accurate parameter combinations from each group. Thus, Figure 8.6 compares the raw network output (labelled as *esn.raw*), selected as the *taps mean* data processing method, against the *combined* feature construction method, labelled *esn.constructed*.

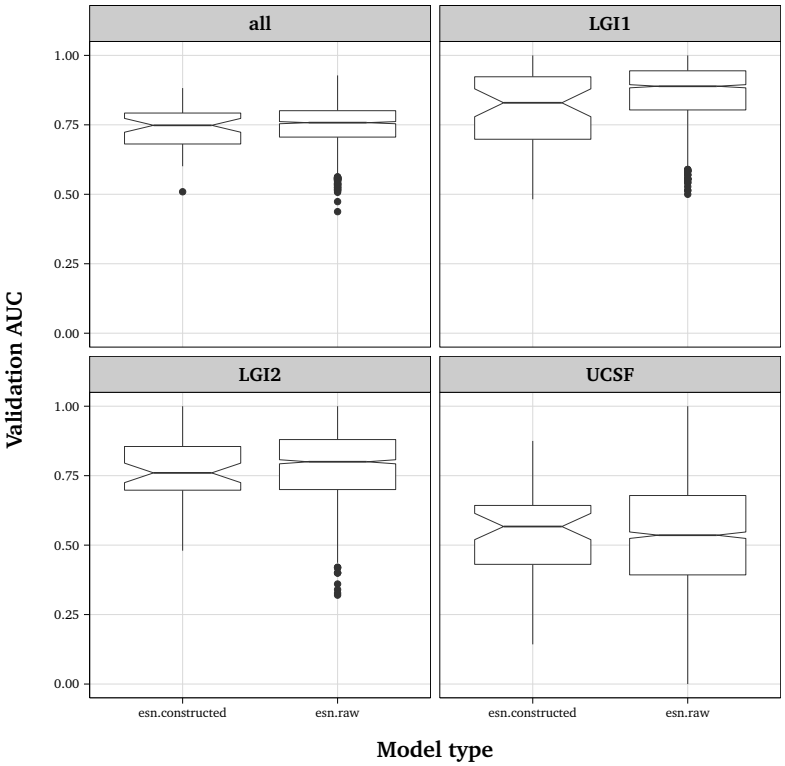


Figure 8.6: Comparison of raw ESN classifiers and ensembles modeling the constructed features

Interestingly, as with the expression tree classifiers, there is little observable difference in predictive power between using the models individually, or combining their outputs into new data sets of constructed features, however the matched Wilcoxon test indicates that there is a statistically significant difference at the 5% level. The lack of improvement offered by the constructed data sets could be owing to the fact that the models analysing the constructed features are overfitting to the training set, as the networks constructing the attributes have been selected according to their prowess on these data samples. By further training a model on these selected features it is likely that some generalising ability is being lost, similarly to how the Evolved Vote Aggregator (EVA) ensembles exhibited overfitting as a result of being trained twice on the same samples. To further compare the modeling techniques, each type can be summarised by their mean AUC, which is 0.731 for the *esn.raw* networks, and 0.714 for *esn.constructed*, indicating that using the network as a classifier individually produces more accurate classification models than incorporating a secondary feature construction phase, despite requiring less computational time.

8.5 Comparison to Temporal Expression Tree Classifiers

Table 8.2: Overall summary of PD movement data modeling approaches

Learning algorithm	Mean AUC	SD
BK.features	0.763	0.187
BK.features.ext	0.754	0.199
esn.raw	0.731	0.176
gp.constructed	0.727	0.188
tetc.mean	0.722	0.185
esn.constructed	0.714	0.162

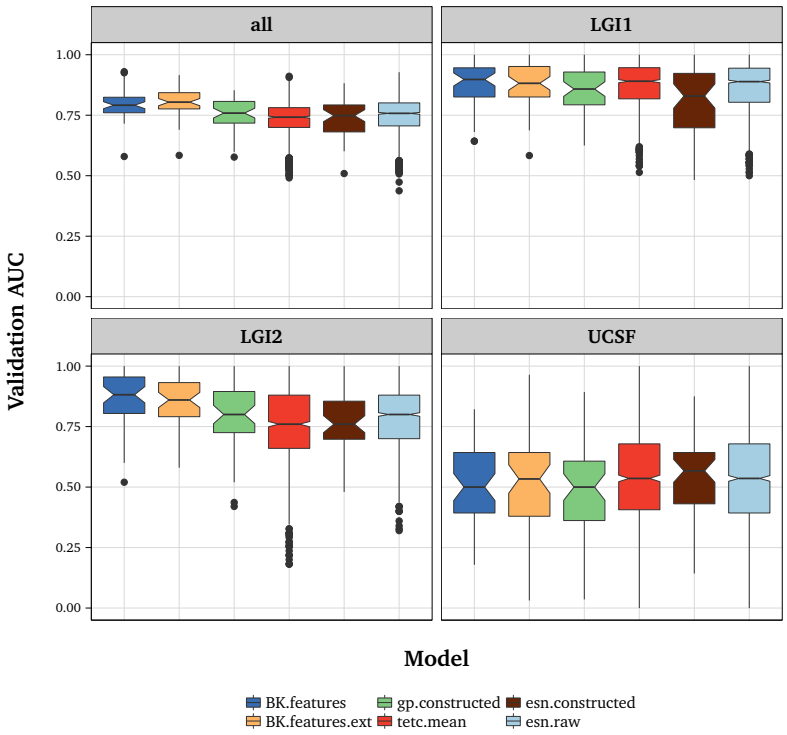


Figure 8.7: Comparison of the full range of modeling techniques on the PD identification data

To place the ESN results in context with the classification approaches seen in previous chapters, Table 8.2 displays the average AUC scores for all of the main data processing methods observed so far, while Figure 8.7 shows the distribution of AUC scores per medical centre. As the overall Friedman test produced a significant p-value at the 0.1% level, post-hoc pairwise Nemenyi tests were run to further investigate the source of differences between models, the results of which are shown in Table 8.3.

On the whole, the ESNs perform moderately better than the TETCs individually, as well as

Table 8.3: Post hoc Nemenyi test p-values for the simulation

	BK.features	*BK.features.ext*	*gp.constructed*	*tetc.mean*	*esn.constructed*
BK.features.ext	1				
gp.constructed	$p < 0.1\%$	$p < 0.1\%$			
tetc.mean	$p < 0.1\%$	$p < 0.1\%$	0.78		
esn.constructed	$p < 0.1\%$	$p < 0.1\%$	0.44	0.99	
esn.raw	$p < 0.1\%$	$p < 0.1\%$	1	0.9	0.61

ensembles analysing constructed features from these trees. However, the difference is not as large as could be expected given that ESNs incorporate short term memory by means of a time delay, and indeed is not statistically significant. Both of these techniques, along with the combined feature design data set approaches, are significantly less accurate than modeling the bradykinesia features, although the difference is not so vast that using these fully objective classifiers would be placing oneself at a significant disadvantage when selecting a model for a direct application. The primary difference in overall accuracy between the classifiers analysing raw data and those receiving the bradykinesia measures is the relative weakness of the raw expression tree and ESNs models on samples recorded at *LGI2*, which then compromises the score on the *all* data set.

8.6 Conclusions

This chapter has investigated several aspects of classifying movement data using ESNs, from considerations to make when pre-processing the data, to means of producing the final model, in addition to discussing the impact of objectively constructing a new feature set. As with the raw expression tree classifiers seen in Chapter 7, forming a secondary data set combining the outputs of multiple networks being input the time series does not tend to provide significant benefits over using the networks individually. This could be partly due to overfitting, as the classifiers are selected to form the feature construction data set by means of their training set score, thereby providing little guarantee that the formed data attributes are indicative of global trends in the unseen test data as well as the training samples.

Despite having recurrent connections and thus incorporating the previous reservoir state at each update, the ability of ESNs to model temporal trends in the data is not largely exploited since the overall accuracy is only moderately greater than that from static expressions formed using GP, although the vastly reduced training time of ESNs when compared to GP should be taken into account for any applied modeling. Further work could investigate alternative means

of classifying temporal data using ESNs to take advantage of the recurrency to provide greater discriminatory ability. Data in the form of a periodic signal allowed for simple segmentation into its constituent cycles, which was shown to provide greater accuracy than presenting the position signal to the network sample-by-sample, despite the former approach not utilising the short term memory of the ESN as much as the latter.

Useful contributions to knowledge have been made by the analysis of different aspects of the network configuration for a classification application. Calculating an aggregate state vector for the overall time series was shown to be most effective when using the mean of all the state vectors—as suggested by Lukoševičius (2012)—in contrast to utilising the reservoir state at the final timestep, as was implemented by Verplancke et al. (2010). This difference could explain the results found by the latter, where ESNs were outperformed by Support Vector Machines (SVMs) and Naïve Bayes (NB) classifiers. This work also marks the first occasion in which ESNs (and the greater class of RNNs) have been applied to diagnosing PD from movement data.

While the accuracy of ESNs was not significantly larger than that from the static TETC approach for its additional recurrent connections, this classification model is the overall highest scoring objective approach, and therefore would be recommended for a time series classification application where domain knowledge is not available. There is scope for further experimentation with the ESN learning algorithm to produce better fitting models, including incorporating leaky integrator neurons and suitable bias terms. The relatively poor discriminatory ability achieved by the *last single* networks—which in theory are provided the greatest opportunity to model the finger tapping dynamics—is perhaps surprising, but future work could investigate why this is the case.

Chapter 9

Summary and Conclusions

Contents

9.1 Rationale and Work Conducted

Parkinson's Disease (PD) is an extremely debilitating condition which affects both a person's motor skills and cognitive functionality, causing a dramatic impact upon their standard of living. It is becoming increasingly common in societies with ageing populations, but as of now there is no cure, only treatment to manage the crippling symptoms. As a result, a rapid and accurate diagnosis is crucial to help provide the most effective treatment pathway possible. However, currently in the United Kingdom (UK) there is a lack of specialist care with a large regional discrepancy in the availability of primary care services, furthermore, it is estimated that up to 25% of PD cases are misdiagnosed by general neurologists. Therefore, an accurate, early, and fully objective means of diagnosis would greatly improve the standard of care for

sufferers of PD and enable a better quality of life, in addition to reducing the cost to the healthcare system.

Advances in transducer technology have enabled the production of wearable, highly precise, and non-invasive position sensors. Such technology can be used to provide a detailed insight into the kinematics of Parkinsonian motor disorders, providing a rich collection of movement data from which diagnostic predictions can be made. Current predictive modeling techniques, however, necessitate input data to be in the form of discrete, uniformly sized feature vectors, containing as few attributes as possible for maximum efficacy. Position sensors typically update at frequencies of 60Hz or greater, meaning that even short recordings produce high-dimensional time series data to be classified. The standard approach with such data is to reduce the dimensionality by extracting a discrete set of summary measures; in many cases—such as with medical data—this process requires expert domain knowledge.

A large, international, multi-site study was run, providing a vast amount of positional data from both PD patients and healthy controls performing a simple, yet discriminatory, movement task—finger tapping. Under expert medical guidance, a series of seven movement disorder markers were extracted from the raw data, ready to be modeled.

Ensemble classifiers have become a popular and widely researched area in machine learning owing to their ability to form accurate and robust predictive models; in particular they have been shown to offer an advantage over individual classifiers on data sets the size of those produced from typical medical studies. They require a pool of individually accurate, yet diverse, base classifiers to be combined into a single predictive model. Evolutionary Algorithms (EAs) provide a natural fit as a tool in ensemble building, since they train a collection of individuals and provide a relatively thorough global search, thereby increasing the probability of finding diverse members. Current approaches in the literature, however, have typically involved using Multi-Objective Evolutionary Algorithms (MOEAs) to explicitly balance the accuracy-diversity trade-off of ensemble formation over using standard single objective EAs, resulting in a more complex and computationally expensive learning algorithm than would be obtained otherwise. While such approaches have produced accurate models, they have not commonly been implemented by the machine learning community as a whole, perhaps partly due to their complex nature. In this thesis, a thorough investigation of the suitability of single objective EAs for ensemble building was carried out, considering multiple aspects of the process along the way. The amount of diversity required for successful ensemble fitting was explored in several areas, including: the selection of the base classifier architecture, the use of niching

breeding operators, the manner in which the ensemble members are selected from the final population, and the use of secondary EAs to further train the ensemble voting scheme. This approach of using niching to augment single objective EAs was shown to produce competitive results to ensembles evolved using more complex MOEA schemes.

After showcasing the improvements offered by using EAs for ensemble formation on a set of benchmarks problems, they were applied to the PD finger tapping movement disorder markers. The use of expression trees as the base classifiers was shown to provide a simple means of visualising the discriminatory ability of each movement component due to the inherent feature selection provided by evolved trees. This is an important consideration for applied models, as it provides useful feedback to clinical neurology researchers, enabling them to compare what they consider to be the most discriminatory aspects of a finger tapping cycle with those features deemed important by the classifier. This approach was extended to distinguish between varying levels of cognitive impairment based purely on movement characteristics.

While this approach was successful, the requirement of domain knowledge to reduce the raw time series data into a set of feature vectors was a limitation to providing the overall aim of a fully objective diagnostic aid. Thus, an algorithm to perform this feature design process was sought. EAs again provided a natural fit for these criteria, as they are ideally suited to applications where no domain knowledge is available and the set of potential solutions is too large for a manual search. The periodic nature of the finger tapping separation waveforms also helped reduce the potential solutions to this problem by initially reducing the dimensionality to the number of tap cycles present in the recording.

A Genetic Programming (GP) approach for feature design was then implemented using arithmetic expression trees to provide each tap with a score based on the shape of the separation waveform, before aggregating into a single continuous value for each recording. The EA evolved a population of such trees to optimise discriminatory ability of these designed features. Alongside using these Temporal Expression Tree Classifiers (TETCs) as individual classifiers, a secondary data set was formed from the outputs of a variety of TETCs employing different aggregation functions, however, this approach offered no significant benefit for its additional complexity.

Since the TETC approach lacked an awareness of the temporal aspect of the finger tapping recordings, a method was sought which would be able to exploit this additional dimension. Echo State Networks (ESNs) were chosen for this purpose owing to their recurrent nature

without requiring a convoluted training method, as is required for standard time-delay Recurrent Neural Networks (RNNs). Owing to the relatively small body of research on applying ESNs to classification of time series data, an investigation was carried out to determine the most effective modeling configuration, in addition to classifying the same data set as used by the TETC approach allowing for a direct comparison.

9.2 Conclusions

Thorough empirical testing of each step in producing an objective diagnosis of PD has resulted in an understanding of the considerations to be made when developing such a model, and has produced the following conclusions.

9.2.1 Evolutionary Algorithms can provide an appropriate means for ensemble building

Many aspects of the ensemble building process have been considered in order to produce an optimal model with simple modifications to the standard EA approach. In particular, the application of niching algorithms was shown to help develop higher performing classifiers than those built from standard EAs alone. However, these results were not true for all of the niching methods assessed, reinforcing the concept of the accuracy-diversity trade-off that has been previously identified in the ensemble learning literature. The more effective algorithms were those which maintained a focus on both providing an explorative search and preserving highly fit candidate solutions; approaches which concentrated their efforts on just one of these areas were typically less accurate. The ensemble building research also highlighted the observation that when using a quantifiable means of similarity between two classifiers, the choice of measure is not as impactful as the mere fact that it is being considered, again, supporting findings from the literature.

The notion of further training the ensemble was shown to aid classification accuracy, however, as anticipated, it had a tendency to overfit to the training set. If this issue could be overcome, it is expected that ensemble accuracy will further increase. A surprising finding was that trained ensembles with their votes combined linearly were comparable to evolved non-linear voting functions, this has implications for real world practice where a small increase in accuracy is typically not worth the large coinciding increase in model complexity. This is especially relevant for medical practice where interpretability is a key factor in classifier

selection. Overall, these evolved ensembles performed well on a variety of benchmark data sets, out performing standard single classifier approaches from traditional predictive modeling literature and resulting in similar accuracies to more involved MOEA-based ensembles from the literature. However, they were not as accurate as state of the art learning algorithms, including non-linear Support Vector Machines (SVMs), AdaBoost, and Random Forests.

9.2.2 Ensemble classifiers applied to movement data can be a useful diagnostic aid for Parkinson's Disease

Applying the evolved ensembles to diagnosing PD from movement disorder markers extracted from the finger tapping data produced encouraging results, with an overall median Area Under Curve (AUC) of 0.80, competitive with the estimated neurologist diagnostic accuracy of 75%. This accuracy was improved upon by models of data recorded at two of the three international sites, stressing the importance of consistency when recording raw data, as the average AUC dropped from 0.85 to only slightly better than a random guess at the remaining location. This indicates that a subtle difference in protocol was implemented in the two centres, resulting in vastly different discriminatory abilities despite recording the same simple motion. Subsequent analysis indicated that classifiers were overfitting to a far greater degree on the under-performing data set—*UCSF*—than on the others. To reduce the error caused from variance, ensembles trained on the combined data set were shown to generalise better and produce more accurate predictions on the *UCSF* data set than those specifically trained on this data. This has important implications for medical applications of predictive modeling, where a variety of data recorded under different circumstances is typically available. By incorporating as many samples as possible into the training set—regardless of their individual diagnostic ability—more generalised and robust models can be formed.

While the ensembles were not as successful at modeling differences in PD cognitive impairment levels, there were interesting findings resulting from this work. The model accuracies between the different levels of cognitive functionality reinforced the gulf in impairment between PD patients diagnosed as suffering with Mild Cognitive Impairment (MCI) and Parkinson's Disease with Dementia (PDD), meanwhile the difference between no cognitive impairment and MCI was rather more subdued with models unable to improve upon a random guess.

The use of expression trees to model the PD markers provided an inherent method of feature selection, resulting in a visualisation of which movement components were determined to be most PD indicative, thereby providing useful feedback to clinical neurology researchers.

This is in contrast to Artificial Neural Networks (ANNs) which are commonly employed as the base classifier in evolved ensembles, and operate in more of a black box manner. This ability is extremely important for applied modeling, where sometimes understanding how the predictions are made is as important as having the most accurate classifier possible. This is often at odds with the typical approach taken by Evolutionary Computation (EC) researchers, where the performance of a system is tweaked to produce minute improvements at the cost of program complexity. It is hoped that this thesis will contribute the importance of considering classifier usability to EC researchers working on predictive modeling applications.

9.2.3 Genetic Programming can successfully model periodic time series data

An extension to a preliminary feature design approach using GP was developed—called Temporal Expression Tree Classifiers (TETCs)—to further exploit the periodic nature of the finger tapping movement task to provide one level of dimensionality reduction, before reducing into a single summary value by aggregating the output of each tap cycle from the evolved function. The inputs to the expression trees comprised ratios of the tap cycle, producing models which discriminated based on the shape of the tapping waveform. TETCs were successful in developing models that were competitive with the movement disorder markers approach, albeit not quite as accurate. However, they did offer some advantages over the features extracted with expert medical knowledge, as the finalised classifiers offered a simple visualisation of how the predictions were formed, since the finger tapping cycles were scored independently and thereby allowing them to be ranked in terms of predicted likelihood of belonging to a PD patient. Furthermore, the tap window offsets were evolved as inputs to the expression trees, allowing for an objective determination of the most discriminatory phases of the movement.

It was discovered that the expression trees typically looked at the separation profile of each tap, identifying smooth sinusoids as being most indicative of healthy subjects, while cycles with less well defined features were identified as more likely belonging to PD patients. Combining multiple designed features into a single data set—inspired by the ensemble research—resulted in a slight, but statistically insignificant, increase in predictive power over using the TETCs individually, but still not as accurate as models formed on the bradykinesia markers.

9.2.4 Echo State Networks provide a simple means for classifying time series

Overall, it was discovered that utilising a dynamical system to model movement disorders offered moderate advantages over the static expression tree approach, although perhaps not as large as could be expected given that the ESNs have access to an additional dimension of the positional data. Several aspects of implementing ESNs for classification models were investigated, including the manner in which the aggregate activation states were calculated for each input time series, in addition to the effect of preprocessing the data into discrete tap cycles. It was found that exploiting the periodic nature of the finger tapping data to form a simple means of dimensionality reduction produced more accurate networks than by passing in the time series sample by sample, despite the latter approach allowing for the temporal aspect to be modeled more closely.

The ESNs which formed their outputs from the mean tap score can be viewed as operating similarly to the TETCs, being passed the data tap-by-tap and forming an aggregate state output vector from the average state across the timesteps. However, the expression tree approach is static, the order in which the tap separation traces are passed into the function has no bearing on the overall score. In contrast, the average state of the ESN is a function of all previous states and so the time dimension is taken into account when modeling the movement data. For an applied model, however, the computational time must be considered, which is significantly lower when training ESNs compared to an evolutionary run. Combining designed features from multiple networks was not shown to aid diagnostic abilities for the ESN approach.

9.2.5 Evolutionary Algorithms have limitations as practical learning algorithms

There are several potential reasons for the lack of uptake of EAs as a learning algorithm by the machine learning community, which highlights that while they provide a valuable technique for many applications, they are limited when it comes to standard predictive modeling tasks. For example, there is no standard implementation of an EA as there are multiple areas where the evolutionary search can be tweaked, such as the individual representation and the selection, mutation, crossover, and replacement operators. Furthermore, each of these areas contains multiple hyper-parameters to be optimised. While experienced EA practitioners have an intuition of appropriate parameter values, the flexibility can prove overwhelming to machine learning experts who are more acclimatised to learning algorithms that contain only a handful of tuning parameters.

There is another, more practical downside to having such a flexible algorithm for predictive modeling: to select the optimal model for each problem, multiple candidate models are built using a variety of hyper-parameter values. This is often repeated over various resampling strategies such as cross-validation or bootstrapping to reduce the effects of bias when performing model selection; however, applying a similar methodology to Evolutionary Algorithms can prove extremely computationally expensive—even compared to the most complex statistical learning algorithms such as boosting—due to the evaluation of many candidate models which are later discarded. Indeed, many EC studies do not optimise the algorithm for each run due to the large amount of time this would require. Furthermore, the stochastic optimisation itself offers no guarantee that optimal parameter values will be located. While EAs have not found regular use as a training algorithm for the underlying classifiers, they have been used successfully as feature selection algorithms in the form of simple bit string Genetic Algorithms (GAs). Here there are fewer hyper-parameters to tune than for a full GP run and there is a natural phenotypical representation, maintaining accessibility.

9.3 Hypothesis Revisited

The work described in this thesis has investigated the hypothesis that

> *Computational Intelligence techniques offer an objective and accurate diagnosis of Parkinson's Disease from simple movement tasks measured during conventional clinical assessment.*

This has been achieved through a progression of experiments researching multiple aspects of the hypothesis, providing an understanding of the considerations when constructing classification models of time series data using Computational Intelligences (CIs). Along the way, many useful findings have been uncovered which will prove beneficial for future work in this field.

9.4 Contributions

The work described in this thesis has made contributions to the literature in various fields of both CI and machine learning, alongside providing valuable information about the mechanics of Parkinsonian movement disorders from the clinically motivated investigations into this disease. The primary contribution is the demonstration that CI-based feature design strategies

can be used to independently form measures from time series with similar levels of discriminatory ability to models built using features extracted with expert clinical PD knowledge. Two different CI techniques have been applied for this purpose: GP and ESNs.

A survey of evolutionary ensemble building approaches highlighted a lack of research into the suitability of standard, single-objective algorithms for this purpose, with more complex MOEAs typically being employed instead. A novel evolutionary ensemble algorithm was developed, incorporating niching strategies to enforce diversity amongst the population at the breeding stage, rather than at the fitness evaluation level as has been previously considered. While the issues of ensemble similarity, ensemble voting, hybrid ensembles, and the selection process have been investigated individually in previous research, the work described in Chapter 5 investigated each of these areas at the same time under the context of a single objective EA. The developed algorithm was shown to perform competitively with the more complex MOEA-based approaches from literature, which also include local training in the form of backpropagation.

This work also introduced the extension of a previously developed GP-based feature design strategy for periodic data. The use of multiple aggregation functions allowed the combination of multiple such models, thereby reducing the dimensionality of the time series into a new objectively created feature space in a method taking inspiration from the ensemble research. In this thesis, ESNs have been applied to modeling neurodegenerative diseases for the first time. In particular, such networks were used directly for classification by modeling the desired output label, rather than building models forecasting the movement signal and being separated in model space in the manner of kernel-based techniques. ESNs prove attractive for applied modeling of medical data owing to their rapid training time.

An additional contribution to knowledge stems from analysis of the PD movement data itself, in addition to models formed on it. Various facets of Parkinsonian movement disorders, selected by experts on the disease, have been assessed for their discriminatory properties by a simple analysis of the ensemble classification models. It is hoped that these findings are of some worth to the clinical neurological research community, along with the discovery that the finger tapping protocol of the Movement Disorder Society sponsored revision of Unified Parkinson's Disease Rating Scale (MDS-UPDRS) places a greater emphasis on identifying the sequence effect at the expense of other more discriminatory components (Appendix B). This analysis of the disease has also reinforced the significant impact a severe cognitive deterioration has on motor skills, in the case of PDD patients.

9.5 Further Work

There is great scope to extend the work investigated in this thesis, this section describes the most interesting to the author. The trade-off between accuracy and diversity when building ensemble classifiers is a well known issue, and was highlighted in this work during the training of a pool of base classifiers using niching EAs. There was a significant difference between the various niching algorithms under investigation, with a pattern emerging that those methods which balanced the trade-off were more effective. It would be extremely valuable for the ensemble learning research community to derive an expression relating the quantifiable measures of accuracy and diversity required for optimal ensemble accuracy. In addition, the choice of specific similarity feature was shown to not impact, instead a greater emphasis was placed on the decision to employ similarity measures or just use elitist selection strategies. This phenomenon has been observed previously, but there is still a lack of awareness of whether there is an optimal similarity quantity describing ensemble members.

There is also considerable scope for extending the work on objective classification of the movement data. While ESNs classifiers were shown to be more accurate than using static expression tree functions, the difference was not as significant as could be expected given the ability of the former to identify trends over time. Other approaches could be considered, for example other dynamical systems alongside ESNs, including alternative reservoir computing techniques such as Liquid State Machines (LSMs). Deep learning approaches, such as Deep Belief Networks (DBNs), could potentially offer advantages here as they have demonstrated a strong ability to reduce high-dimensional data for classification purposes, albeit typically applied to image recognition. Another approach to classifying time series data is to provide a custom kernel function to SVMs, allowing for the direct input of raw data instead of discrete summary features. Such a function would require the derivation of a quantifiable similarity measure between two time series; a large number of candidate distance functions can be imagined. A statistical approach could be to calculate the correlation coefficient between the two traces to define their similarity, or to count the number of easily observed features in common such as local optima. From a signal processing perspective there is a variety of ways in which such a kernel function could be implemented. For example, similarity could be measured as a function of the frequency spectrum of each input, such as the Euclidean distance between power levels of various frequency bins. The cross-correlation provides another means for assessing similarity between two signals, although the resultant spectrum would need to be aggregated into a single measure, perhaps by the number or amplitude of peaks.

The concept of forming a new data set comprising objectively constructed features was not shown to offer any advantage over using the classifiers individually, however, further work in this area could look into optimum combination techniques to provide maximum accuracy. This area contains considerable overlaps with ensemble learning, allowing for the adaptation of similar techniques for ensuring diversity. For example, one popular method of maintaining member diversity for ensemble classifiers is to train them on differing subsets of the feature space. Incorporating these principles to objective feature construction could result in classifiers being trained on different components of the time series, for example, one model could be passed the raw position data, while another could receive the trace after undergoing a smoothing process. The first and second derivatives of the position signal offer an alternative perspective of the motion, and could be incorporated into a summary feature set, likewise with rotational data if available. Other areas adapted from ensemble learning could include measuring the correlation between individuals, or using hybrid classifiers; for example, having a feature set comprising attributes constructed using both ESNs and expression trees. However, care must be taken when doing so to reduce overfitting to the training set, as it is believed that this occurred during the feature construction process seen in Chapters 7 and 8.

The methods described in this work have focused on the MDS-UPDRS finger tapping task due to clinical motivations, such recordings have an advantage for objective feature design as the periodic nature allows for simple pre-processing into tap cycles, thereby reducing the dimensionality for free. It would be useful to produce generic classification techniques for all movement data, not just that of a periodic nature. This would enable a composite diagnosis to be formed using multiple movement tasks, thereby providing a more robust model of PD; particularly when combined with the previously described raw data classification techniques. Logistically this would not present too many challenges, as PD patients are already required to undergo a series of movement tasks as part of the MDS-UPDRS and the same non-invasive position sensors could be used. This would greatly facilitate a truly objective and accurate diagnosis of Parkinson's Disease using Computational Intelligence techniques.

Appendix A

Variance of Predictive Models formed using Evolutionary Algorithms

A.1 Introduction

When evaluating the performance of a learning algorithm for producing applied classification models, it is important to obtain an accurate estimate of its generalisation ability to unseen data. The simplest way of assessing generalising capability is by subsetting the data set into *training* and *test* partitions, whereby the learning algorithm fits a model on the former and is evaluated on the latter. As the test observations have not been included in the training phase, evaluation on this subset provides an estimate of the model's predictive skill for future unseen data. However, this technique is not perfect as the selection of samples in the test set provides a source of variance, for example, if the patterns closest to the separation boundary were randomly allocated to the test set then it would result in an optimistic estimated accuracy. Resampling techniques—such as *k*-fold cross-validation and bootstrapping—are popular for model evaluation, as they test the learning algorithm on a range of unseen test sets, reducing the variance of the error due to the sample selection. *k*-fold cross-validation partitions the data into *k* equally sized data *folds*, and iterates through using each fold as the test set and the remaining $k-1$ folds for training. Typical values for *k* are five or ten.

Computational Intelligence (CI) algorithms typically comprise some stochastic nature in their behaviour, commonly in the initial conditions or during the search process itself. For example, backpropagation trained Multi-Layer Perceptrons (MLPs) are initialised randomly before being fitted to data with a deterministic learning algorithm. Evolutionary Algorithms (EAs), however,

include both of these elements, with the initial candidate solutions being constructed randomly and further stochastic nature present during the selection and reproduction phases. As a result, when estimating the generalising capability of classifiers formed using EAs, the practitioner must be aware of this additional source of variation alongside that resulting from the sampling of the data set.

The aim of this investigation is to determine which is the greater source of variation in applying EAs to predictive modeling: resampling of the data set or the intrinsic variation across EA runs. This will provide a greater understanding of the underlying mechanics of using EAs as a learning algorithm and aid the choice of resampling method for model evaluation and selection, establishing whether any modification to the standard cross-validation procedure is required.

A.2 Methodology

Two sets of classification simulations were run, investigating variance in estimated model accuracy resulting from both the resampling method and the stochastic nature of EAs. For both set of experiments, a resampling profile of thirty repeats of stratified 10-fold cross-validation was employed to ensure a robust set of results, as well as being a similar approach to model evaluation as would be used in practice. Formally, there were R_i repeats where $i \in \{1, 2, \ldots, 30\}$ and F_j folds with $j \in \{1, 2, \ldots, 10\}$.

For the simulations investigating variance due to data resampling, the splitting of the data into the ten folds was performed independently for each repeat. A classifier was then trained using EAs and evaluated for each fold of each repeat in the standard cross-validation manner. The random number generator seed used to construct the initial population and form the stochastic breeding operators was kept constant across the repeats for each fold. I.e., $seed(R_1F_j) = seed(R_2F_j) = \ldots = seed(R_{30}F_j)$ for all j. Calculating the variance of the classifier accuracy across the repeats for each fold produced ten measures of variance due to data resampling alone, $var_j = variance(acc(R_1F_j), acc(R_2F_j), \ldots, acc(R_{30}F_j))$ for all j.

To assess the variation of EAs as a learning algorithm, the thirty repeats of the ten fold cross-validation were identical, but the random number generator seed was kept independent, thereby producing a setup where the data in folds $R_1F_j, R_2F_j, \ldots, R_{30}F_j$ are all equal, but the random number generator seeds dictating the evolutionary path are distinct. The variance in classifier accuracy across these folds is thereby solely due to the stochastic aspect of EAs,

again resulting in ten measures of variance for each of the folds.

The particular classification model used in this experiment was Genetic Programming (GP) arithmetic expression trees. These two setups were tested on the ten binary data sets used in Chapter 5, as detailed in Table 5.1. Thus, one hundred variance measures were calculated for each method, comprising the variance across ten folds on ten data sets.

A.3 Results

The distribution of classifier Area Under Curve (AUC) variance scores for each fold across the ten University of California Irvine (UCI) data sets is shown in Figure A.1. The plot highlights that the variance in AUC is significantly greater due to data partitioning than from the stochastic nature of the EA search algorithm. This is a key finding, for any learning algorithm to provide inherent worth it must not produce a large amount of variance in the models it builds. By showing that EAs develop classifiers with lower resulting variance than from data resampling, a practitioner can safely use cross-validation to perform model selection and evaluation, in the same way that it is employed for alternative learning algorithms which either also include stochastic elements—such as MLPs and random forests—or do not.

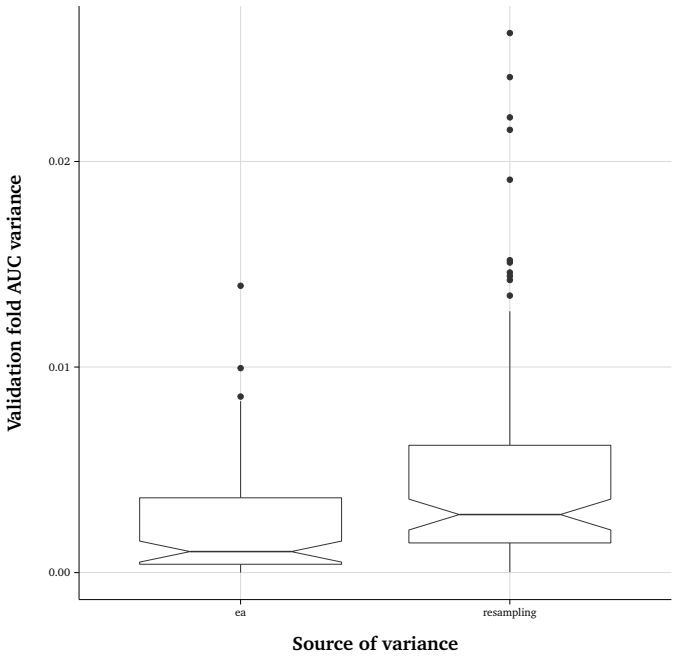


Figure A.1: Comparison of variance sources

A.4 Conclusions

The finding that the variation in classification ability due to the sampling of the data set is greater than that from differences across EA runs has several important implications for predictive modeling research using such techniques. First, it highlights the suitability of using EAs for predictive modeling in general, as although they contain more sources of variation than established techniques such as Support Vector Machines (SVMs), this has minimal impact upon the evaluation of models trained using this method. In addition, when implementing EAs to build a model to be used in an applied setting, the choice of random number generator seed to be used will be largely inconsequential. Furthermore, these results indicate that standard resampling techniques—such as *k*-fold cross-validation—are suitable for evaluating classifiers trained using EAs and that no further action to assess variation resulting from the stochastic nature of EAs is required.

Appendix B

Does the Unified Parkinson's Disease Rating Scale Finger Tapping Task Allow For the Sequence Effect to be Exhibited?

Contents

B.1 Introduction

The primary objective behind recording the *LGI1* data set was to investigate the effectiveness of the Movement Disorder Society sponsored revision of Unified Parkinson's Disease Rating Scale (MDS-UPDRS) finger tapping task in highlighting the sequence effect. The sequence effect is a progressive decline in amplitude and speed of repetitive movements and is one of the cardinal symptoms of Parkinson's Disease (PD); test administrators are instructed to consider it in their composite scoring of the ten taps of the MDS-UPDRS finger tapping assessment—as detailed in Section 6.2.2. Since participants are required to tap as quickly as possible, a cycle of ten finger taps can last a matter of seconds, which may not be long enough for the sequence effect to manifest.

Table B.1: Details of the test subjects used during the Sequence Effect study

Study	Control subjects	PD patients
LGI1	38	49
UCSF	9	25
Total	47	74

To investigate the clinically motivated hypothesis that using a duration of ten taps is not long enough for the sequence effect to be exhibited in PD patients performing finger tapping, a measure representing the sequence effect was derived and compared as a marker of PD for its discriminatory ability at various lengths of tapping tasks.

B.2 Methodology

Data was collected from the full *LGI1* cohort and a subset of the *UCSF* group who had been initially assessed, discarding three controls to form an age-matched data set of forty-seven controls and seventy-four PD patients (Table B.1). The participants were recorded finger tapping with both dominant and non-dominant hands, as described in Section 6.2. To establish the impact of the number of tap cycles, the seven summary measures representing components of bradykinesia and the sequence effect previously seen in Chapter 6 were calculated after varying numbers of complete tap cycles.

Each of the movement disorder markers described in Table 6.2 was calculated at different test durations, measured by number of complete tap cycles as instructed by the MDS-UPDRS guide rather than time; the durations chosen were ten, twenty, thirty, and forty taps. To measure how strong an indicator of bradykinesia each feature was, Receiver Operating Characteristics (ROC) characteristics were calculated with the Area Under Curve (AUC) measuring the probability that a randomly picked PD recording had a value than a randomly picked control recording. There were two primary objectives: first to establish whether the sequence effect is more distinguishing of PD over ten taps than longer recordings, and to determine whether any other Parkinsonian movement characteristics react similarly to different finger tapping lengths.

Table B.2: Discriminatory ability of the sequence effect over a range of tap lengths

Duration (taps)	PD % fatigue	HC % fatigue	AUC
10	59.38	37.35	0.647
20	60.16	56.63	0.592
30	66.41	57.83	0.612
40	67.19	62.65	0.589

Table B.3: Logistic regression coefficients for a univariate model of *fatigue.sep*

Duration (taps)	Coefficient (odds)		Odds(PD | fatigue = -0.001)
	Intercept	Fatigue	
10	1.58	0.75	1.62
20	1.42	0.63	1.48
30	1.36	0.42	1.48
40	1.37	0.40	1.50

B.3 Results

Table B.2 displays the percentage of PD patients and healthy controls exhibiting decrementing separation (*fatigue.sep* < 0), alongside the AUC of using this measure as a classifier. The proportion of PD patients exhibiting decrementing amplitude increases with the number of taps, although a majority (59%) show signs of it during only the ten taps required by the MDS-UPDRS. The percentage of healthy control subjects showing a decline in successive tap separation also increases with the number of finger tap cycles, but only a minority of the group show signs of it over ten taps, with similar ratios as PD patients showing fatigue at longer durations.

This indicates that the decrementing amplitude evident in both healthy subjects and PD patients at longer test durations is a result of standard physical fatigue, while the same decrease in tap amplitude at shorter test lengths is more largely comprised of the sequence effect, since this phenomenon is observed in a majority of PD patients but only a minority of healthy controls. These results suggest that assessing finger tapping over just ten taps is sufficient for the sequence effect to exhibit itself in PD patients, as at longer durations physical fatigue is an additional contributing factor to decrementing amplitude and can mask the bradykinesia symptoms. This is supported by the AUC score from using the *fatigue.sep* marker as a classifier, whereby the greatest discriminatory ability is found on the data recorded across ten tap cycles.

To further investigate these findings, a univariate logistic regression model was fitted to

the data at each tap duration, implementing *fatigue.sep* as a continuous predictor with the subject type as the dichotomous response. The coefficients for the final models are detailed in Table B.3. The *Fatigue* column shows the odds increase of the sample belonging to a PD patient from a one unit increase in *fatigue.sep*, as anticipated these values are less than one, indicating that an increase in fatigue reduces the odds of the recording belonging to a PD patient. In addition, these odds have an indirect relationship with tap duration. The *Intercept* value represents the odds that a data recording with a *fatigue.sep* value of 0 is from a subject suffering with PD. The intercept coefficients follow the same trend as the AUC values from Table B.2, strengthening these findings. These coefficients are helpful for understanding the data, as they demonstrate that as the tap duration increases, *fatigue.sep* becomes less indicative of a person suffering from PD. However, for this investigation a more useful measure is the probability that a person with a slight decline in amplitude is exhibiting the sequence effect and not physical fatigue. A value representing a slight decrement of amplitude was determined as -0.001, chosen as the maximum, minimum, and median overall values for *fatigue.sep* are 0.081, -0.17, and -0.0014 respectively. The last column of Table B.3 therefore shows the odds that a person with slight decrementing amplitude suffers from PD. It demonstrates that a recording with this level of fatigue is more likely to result from a patient than a control, but these odds decrease with increase tap duration with the biggest change occurring after ten taps. All these results support the conclusion that a test duration of ten taps is most discriminatory for the sequence effect, as measured by the slope of a regression line of the tap separation values.

To compare these findings to other characteristic Parkinsonian movement disorders, Figure B.1 plots four different bradykinesia markers—*mean.max.sep*, *mean.max.speed*, *cov.max.sep*, and *fatigue.sep* representing hypokinesia, bradykinesia, akinesia, and the sequence effect respectively—against the AUC when being used as a univariate model across the range of tap durations. The main finding is that each of these features becomes more discriminatory with longer tap sequences, except for the sequence effect which has been shown to decrease in diagnostic ability with increased recording length. Furthermore, the sequence effect is shown to be the least discriminatory measure overall, even at its maximum AUC over ten taps.

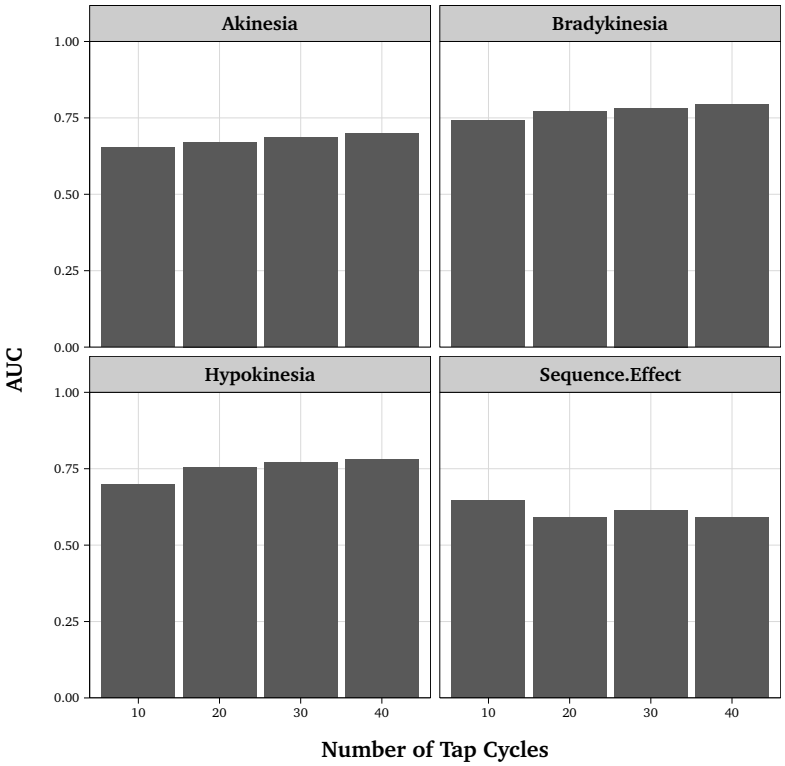


Figure B.1: Discriminatory ability of various bradykinesia measures at different tap lengths

B.4 Conclusions

The results demonstrate that—contrary to initial expectation—a recording of length ten taps is sufficiently long to measure decrementing amplitude to use as a discriminatory factor between PD patients and healthy controls, as patients have a greater element of fatigue even in short sequences of ten taps, while after longer durations this difference becomes less pronounced. This finding indicates that controls are starting to fatigue as well, although this would most likely be due to physical fatigue, while PD patients also suffer from neurological fatigue which manifests itself as the sequence effect during short repetitive movements. Thus, ten taps is sufficient to observe the sequence effect and a longer test length would actually be disadvantageous for this purpose.

However, the MDS-UPDRS finger tapping score is a composite of various factors of Parkinsonian movement disorders, including the sequence effect. Other movement components which test administrators are told to consider—including amplitude, speed, and hesitations—actually offer greater diagnostic power as the test length increases. Thus, it is worth considering whether a duration of ten taps is sufficient for the MDS-UPDRS task to highlight movement

disorders characteristic of PD, as using a longer test length would allow for greater classification ability since the sequence effect is the sole component which benefits from a shorter movement duration. In addition, the sequence effect was the least discriminatory feature of the four that were analysed, suggesting that by increasing the duration and instructing test administrators to not assess decrementing amplitude, greater diagnostic ability can be drawn from the MDS-UPDRS finger tapping task.

Glossary

Medical Terms

AD Alzheimer's Disease

CDR Clinical Dementia Rating

HD Huntington's Disease

ICU Intensive Care Unit

LGI Leeds General Infirmary

LID Levodopa-induced dyskinesia

MAC Memory and Aging Center

MBRS Modified Bradykinesia Rating Scale

MCI Mild Cognitive Impairment

MDS-UPDRS Movement Disorder Society sponsored revision of Unified Parkinson's Disease Rating Scale

MDS Movement Disorder Society

MMSE Mini-Mental State Examination

MoCA Montreal Cognitive Assessment

PD Parkinson's Disease

PDD Parkinson's Disease with Dementia

PSP Progressive Supranuclear Palsy

ROCF Rey-Osterrieth Complex Figure

SFVA San Francisco Veterans Association

SPECT Single-Photon Emission Computed Tomography

SVM Support Vector Machine

UCSF University of California, San Francisco

UHDRS Unified Huntington's Disease Rating Scale

UKBBDC UK Parkinson's Disease Society Brain Bank Diagnostic Criteria

UPDRS Unified Parkinson's Disease Rating Scale

UK United Kingdom

USA United States of America

Computational Intelligence Terms

ADALINE Adaptive Linear Neuron

ABC Artificial Bee Colony

ABN Artificial Biochemical Network

ACO Ant Colony Optimisation

AIS Artificial Immune System

AI Artificial Intelligence

ANN Artificial Neural Network

AUC Area Under Curve

CART Classification and Regression Trees

CGP Cartesian Genetic Programming

CI Computational Intelligence

CTRNN Continuous Time Recurrent Neural Network

DBN Deep Belief Network

DC Deterministic Crowding

DIVACE Diverse and Accurate Ensembles

DSP Digital Signal Processing

DTW Dynamic Time Warping

EA Evolutionary Algorithm

EC Evolutionary Computation

EANN Evolutionary Artificial Neural Network

EENCL Evolutionary Ensembles with Negative Correlation Learning

EM Electro-Magnetic

EP Evolutionary Programming

ES Evolutionary Strategy

ESN Echo State Network

EVA Evolved Vote Aggregator

FFT Fast Fourier Transform

GA Genetic Algorithm

GLM General Linear Model

GP Genetic Programming

GPU Graphics Processing Unit

GRN Gene Regulatory Network

FN False Negative

FP False Positive

FPR False Positive Rate

kNN k-Nearest Neighbours

LCS Learning Classifier System

LDA Linear Discriminant Analysis

LMS Least Mean Squares

LSM Liquid State Machine

LSTM Long Short-Term Memory

MAUC Multi-class Area Under the Curve

MLP Multi-Layer Perceptron

MOEA Multi-Objective Evolutionary Algorithm

MPANN Memetic Pareto Artificial Neural Network

MSE Mean Square Error

MV Majority Vote

NB Naïve Bayes

NCL Negative Correlation Learning

NEAT NeuroEvolution of Augmenting Topologies

PADO Parallel Architecture Discovery and Orchestration

PC Probabilistic Crowding

PFC Pairwise Failure Crediting

PSO Particle Swarm Optimisation

RMSE Root Mean Square Error

RNN Recurrent Neural Network

ROC Receiver Operating Characteristics

RTS Restricted Tournament Selection

SCEA Species Conserving Evolutionary Algorithm

SCGA Species Conserving Genetic Algorithm

SLP Single-layer Perceptron

TETC Temporal Expression Tree Classifier

TN True Negative

TP True Positive

TPR True Positive Rate

TNR True Negative Rate

TSF Time Series Forest

UCI University of California Irvine

WMV Weighted Majority Vote

Bibliography

Abbass, H. A. (2001), A memetic pareto evolutionary approach to artificial neural networks, *in* 'AI 2001: Advances in Artificial Intelligence', Springer, pp. 1–12.

Abbass, H. A. (2003), Pareto neuro-evolution: Constructing ensemble of neural networks using multi-objective optimization, *in* 'Evolutionary Computation, 2003. CEC'03. The 2003 Congress on', Vol. 3, IEEE, pp. 2074–2080.

Agostino, R., Berardelli, A., Currà, A., Accornero, N. & Manfredi, M. (1998), 'Clinical impairment of sequential finger movements in Parkinson's disease', *Movement disorders* **13**(3), 418–421.

Agostino, R., Berardelli, A., Formica, A., Accornero, N. & Manfredi, M. (1992), 'Sequential arm movements in patients with Parkinson's disease, Huntington's disease and Dystonia', *Brain* **115**(5), 1481–1495.

Agostino, R., Curra, A., Giovannelli, M., Modugno, N., Manfredi, M. & Berardelli, A. (2003), 'Impairment of individual finger movements in Parkinson's disease', *Movement disorders* **18**(J5), 560–565.

All Party Parliamentary Group for Parkinson's Disease (2009), 'Please mind the gap: Parkinson's disease services today', `http://www.parkinsons.org.uk/sites/default/files/appg_report_please_mind_the_gap.pdf`. Accessed: 2015-09-03.

Allen, R. B. & Kamm, C. A. (1991), A recurrent neural network for word identification from continuous phoneme strings, *in* 'Advances in Neural Information Processing Systems', pp. 206–212.

Aly, N., Playfer, J., Smith, S. & Halliday, D. (2007), 'A novel computer-based technique for the assessment of tremor in Parkinson's disease', *Age and ageing* **36**(4), 395–399.

Alzheimer's Association (2012), 'Treatments for Alzheimer's disease', http://www.alz.org/alzheimers_disease_treatments.asp.

Alzheimer's Society (2012), 'What is Alzheimer's disease?', http://www.alzheimers.org.uk/site/scripts/documents_info.php?documentID=100.

Angeline, P. J., Saunders, G. M. & Pollack, J. B. (1994), 'An evolutionary algorithm that constructs recurrent neural networks', *Neural Networks, IEEE Transactions on* **5**(1), 54–65.

Bache, K. & Lichman, M. (2013), 'UCI machine learning repository'.
URL: *http://archive.ics.uci.edu/ml*

Bagnall, A., Bostrom, A., Large, J. & Lines, J. (2016), 'The great time series classification bake off: An experimental evaluation of recently proposed algorithms. extended version', *arXiv preprint arXiv:1602.01711* .

Bagnall, A., Davis, L. M., Hills, J. & Lines, J. (2012), Transformation based ensembles for time series classification., *in* 'SDM', Vol. 12, SIAM, pp. 307–318.

Banzhaf, W. (2003), Artificial regulatory networks and genetic programming, *in* 'Genetic programming theory and practice', Springer, pp. 43–61.

Barricelli, N. A. et al. (1954), 'Esempi numerici di processi di evoluzione', *Methodos* **6**(21-22), 45–68.

Bellman, R. (1961), *Adaptive control processes: a guided tour*, Vol. 4, Princeton University Press.

Benecke, R., Rothwell, J., Dick, J., Day, B. & Marsden, C. (1987), 'Disturbance of sequential movements in patients with Parkinson's disease', *Brain* **110**(2), 361–379.

Bennett, D., Shannon, K., Beckett, L., Goetz, C. & Wilson, R. (1997), 'Metric properties of nurses' ratings of Parkinsonian signs with a modified Unified Parkinson's Disease Rating Scale', *Neurology* **49**(6), 1580–1587.

Berardelli, A., Rothwell, J. C., Thompson, P. D. & Hallett, M. (2001), 'Pathophysiology of bradykinesia in Parkinson's disease', *Brain* **124**(11), 2131–2146.

Bertram, L., Tanzi, R. E. et al. (2005), 'The genetic epidemiology of neurodegenerative disease', *Journal of Clinical Investigation* **115**(6), 1449–1457.

Betarbet, R., Sherer, T. B., MacKenzie, G., Garcia-Osuna, M., Panov, A. V. & Greenamyre, J. T. (2000), 'Chronic systemic pesticide exposure reproduces features of Parkinson's disease', *Nature neuroscience* **3**(12), 1301–1306.

Bhowan, U., Johnston, M. & Zhang, M. (2011), Evolving ensembles in multi-objective genetic programming for classification with unbalanced data, *in* 'Proceedings of the 13th annual conference on Genetic and evolutionary computation', ACM, pp. 1331–1338.

Bhowan, U., Johnston, M., Zhang, M. & Yao, X. (2013), 'Evolving diverse ensembles using genetic programming for classification with unbalanced data', *Evolutionary Computation, IEEE Transactions on* **17**(3), 368–386.

Bhowan, U., Johnston, M., Zhang, M. & Yao, X. (2014), 'Reusing genetic programming for ensemble selection in classification of unbalanced data', *Evolutionary Computation, IEEE Transactions on* **18**(6), 893–908.

Bonato, P., Sherrill, D. M., Standaert, D. G., Salles, S. S. & Akay, M. (2004), Data mining techniques to detect motor fluctuations in Parkinson's disease, *in* 'Engineering in Medicine and Biology Society, 2004. IEMBS'04. 26th Annual International Conference of the IEEE', Vol. 2, IEEE, pp. 4766–4769.

Bradley, A. P. (1997), 'The use of the area under the ROC curve in the evaluation of machine learning algorithms', *Pattern recognition* **30**(7), 1145–1159.

Brameier, M. F. & Banzhaf, W. (2007), *Linear genetic programming*, Springer.

Breiman, L. (1996*a*), 'Bagging predictors', *Machine learning* **24**(2), 123–140.

Breiman, L. (1996*b*), 'Bias, variance, and arcing classifiers'.

Breiman, L. (2001), 'Random forests', *Machine learning* **45**(1), 5–32.

Breiman, L., Friedman, J., Stone, C. J. & Olshen, R. A. (1984), *Classification and regression trees*, CRC press.

Bridle, J. S. (1990), Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition, *in* 'Neurocomputing', Springer, pp. 227–236.

Brooks, D. J. (2008), 'Optimizing levodopa therapy for Parkinson's disease with Levodopa/Carbidopa/Entacapone: implications from a clinical and patient perspective', *Neuropsychiatric disease and treatment* **4**(1), 39.

Brown, G., Wyatt, J., Harris, R. & Yao, X. (2005), 'Diversity creation methods: a survey and categorisation', *Information Fusion* **6**(1), 5–20.

Burrows, T. & Niranjan, M. (1994), The use of recurrent neural networks for classification, *in* 'Neural Networks for Signal Processing [1994] IV. Proceedings of the 1994 IEEE Workshop', IEEE, pp. 117–125.

Camicioli, R., Grossmann, S. J., Spencer, P. S., Hudnell, K. & Anger, W. K. (2001), 'Discriminating mild Parkinsonism: methods for epidemiological research', *Movement disorders* **16**(1), 33–40.

Chahine, L. M. & Stern, M. B. (2011), 'Diagnostic markers for parkinson's disease', *Current opinion in neurology* **24**(4), 309–317.

Chandra, A. & Yao, X. (2004), DIVACE: Diverse and accurate ensemble learning algorithm, *in* 'Intelligent Data Engineering and Automated Learning–IDEAL 2004', Springer, pp. 619–625.

Chandra, A. & Yao, X. (2006*a*), 'Ensemble learning using multi-objective evolutionary algorithms', *Journal of Mathematical Modelling and Algorithms* **5**(4), 417–445.

Chandra, A. & Yao, X. (2006*b*), 'Evolving hybrid ensembles of learning machines for better generalisation', *Neurocomputing* **69**(7), 686–700.

Chatzidimitriou, K. C. & Mitkas, P. A. (2010), A neat way for evolving echo state networks., *in* 'ECAI', pp. 909–914.

Chee, R., Murphy, A., Danoudis, M., Georgiou-Karistianis, N. & Iansek, R. (2009), 'Gait freezing in Parkinson's disease and the stride length sequence effect interaction', *Brain* **132**(8), 2151–2160.

Chen, H., Tang, F., Tino, P. & Yao, X. (2013), Model-based kernel for efficient time series analysis, *in* 'Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining', ACM, pp. 392–400.

Chen, H. & Yao, X. (2009), 'Regularized negative correlation learning for neural network ensembles', *Neural Networks, IEEE Transactions on* **20**(12), 1962–1979.

Chen, H. & Yao, X. (2010), 'Multiobjective neural network ensembles based on regularized negative correlation learning', *Knowledge and Data Engineering, IEEE Transactions on* **22**(12), 1738–1751.

Chong, F. S. & Langdon, W. B. (1999), Java based distributed genetic programming on the internet., *in* 'GECCO', Citeseer, p. 1229.

Connor, J. T., Martin, R. D. & Atlas, L. E. (1994), 'Recurrent neural networks and robust time series prediction', *Neural Networks, IEEE Transactions on* **5**(2), 240–254.

Cortes, C. & Vapnik, V. (1995), 'Support-vector networks', *Machine learning* **20**(3), 273–297.

Cox, D. R. (1958), 'The regression analysis of binary sequences', *Journal of the Royal Statistical Society. Series B (Methodological)* pp. 215–242.

Cramer, N. L. (1985), A representation for the adaptive generation of simple sequential programs., *in* 'ICGA', pp. 183–187.

Dalrymple-Alford, J., MacAskill, M., Nakas, C., Livingston, L., Graham, C., Crucian, G., Melzer, T., Kirwan, J., Keenan, R., Wells, S. et al. (2010), 'The MoCA well-suited screen for cognitive impairment in Parkinson disease', *Neurology* **75**(19), 1717–1725.

Dam, H. H., Abbass, H. A., Lokan, C. & Yao, X. (2008), 'Neural-based learning classifier systems', *Knowledge and Data Engineering, IEEE Transactions on* **20**(1), 26–39.

Darwin, C. (1859), 'On the origin of species'.

Dawkins, R. (2006), *The selfish gene*, Oxford university press.

De Jong, K. (1975), 'An analysis of the behavior of a class of genetic adaptive systems', *Ph.D. Thesis, University of Michigan* .

Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. (2002), 'A fast and elitist multiobjective genetic algorithm: NSGA-II', *Evolutionary Computation, IEEE Transactions on* **6**(2), 182–197.

Demšar, J. (2006), 'Statistical comparisons of classifiers over multiple data sets', *The Journal of Machine Learning Research* **7**, 1–30.

Deng, H., Runger, G., Tuv, E. & Vladimir, M. (2013), 'A time series forest for classification and feature extraction', *Information Sciences* **239**, 142–153.

Dietterich, T. G. (2000), Ensemble methods in machine learning, *in* 'Multiple classifier systems', Springer, pp. 1–15.

Dietterich, T. G. & Bakiri, G. (1995), 'Solving multiclass learning problems via error-correcting output codes', *Journal of Artificial Intelligence Research* **2**(263), 286.

Dos Santos, E. M., Sabourin, R. & Maupin, P. (2009), 'Overfitting cautious selection of classifier ensembles with genetic algorithms', *Information Fusion* **10**(2), 150–162.

Draves, S. (2006), 'The electric sheep', *ACM SIGEVOlution* **1**(2), 10–16.

Duda, R. O., Hart, P. E. et al. (1973), *Pattern classification and scene analysis*, Vol. 3, Wiley New York.

Duin, R. P. (2002), The combining classifier: to train or not to train?, *in* 'Pattern Recognition, 2002. Proceedings. 16th International Conference on', Vol. 2, IEEE, pp. 765–770.

Eads, D. R., Hill, D., Davis, S., Perkins, S. J., Ma, J., Porter, R. B. & Theiler, J. P. (2002), Genetic algorithms and support vector machines for time series classification, *in* 'International Symposium on Optical Science and Technology', International Society for Optics and Photonics, pp. 74–85.

Eads, D. R., Williams, S. J., Theiler, J., Porter, R., Harvey, N. R., Perkins, S. J., Brumby, S. P. & David, N. A. (2004), Multimodal approach to feature extraction for image and signal learning problems, *in* 'Optical Science and Technology, SPIE's 48th Annual Meeting', International Society for Optics and Photonics, pp. 79–90.

Edgar, J. A. (2007), MEng Project Report: The application of evolutionary algorithms towards the Diagnosis of Parkinson's Disease, Technical report, Department of Electronics, University of York.

Elman, J. L. (1990), 'Finding structure in time', *Cognitive science* **14**(2), 179–211.

Ericsson, A., Lonsdale, M. N., Astrom, K., Edenbrandt, L. & Friberg, L. (2005), Decision support system for the diagnosis of Parkinson's disease, *in* 'Image Analysis', Springer, pp. 740–749.

Espay, A. J., Beaton, D. E., Morgante, F., Gunraj, C. A., Lang, A. E. & Chen, R. (2009), 'Impairments of speed and amplitude of movement in Parkinson's disease: a pilot study', *Movement Disorders* **24**(7), 1001–1008.

Espay, A. J., Giuffrida, J. P., Chen, R., Payne, M., Mazzella, F., Dunn, E., Vaughan, J. E., Duker, A. P., Sahay, A., Kim, S. J. et al. (2011), 'Differential response of speed, amplitude, and rhythm to dopaminergic medications in Parkinson's disease', *Movement Disorders* **26**(14), 2504–2508.

Fahn, S., Elton, R., Committee, U. D. et al. (1987), 'Unified Parkinson's disease rating scale', *Recent developments in Parkinson's disease* **2**, 153–163.

Faraway, J. J. (2005), *Extending the linear model with R: generalized linear, mixed effects and nonparametric regression models*, CRC press.

Floreano, D., Dürr, P. & Mattiussi, C. (2008), 'Neuroevolution: from architectures to learning', *Evolutionary Intelligence* **1**(1), 47–62.

Frank, E. & Witten, I. H. (1998), 'Generating accurate rule sets without global optimization'.

Freitas, S., Simões, M. R., Alves, L. & Santana, I. (2013), 'Montreal cognitive assessment: validation study for mild cognitive impairment and alzheimer disease', *Alzheimer Disease & Associated Disorders* **27**(1), 37–43.

Freund, Y., Schapire, R. & Abe, N. (1999), 'A short introduction to boosting', *Journal-Japanese Society For Artificial Intelligence* **14**(771-780), 1612.

Freund, Y., Schapire, R. E. et al. (1996), Experiments with a new boosting algorithm, *in* 'ICML', Vol. 96, pp. 148–156.

Fulcher, B. D., Little, M. A. & Jones, N. S. (2013), 'Highly comparative time-series analysis: the empirical structure of time series and their methods', *Journal of The Royal Society Interface* **10**(83), 20130048.

Gabrys, B. & Ruta, D. (2006), 'Genetic algorithms in classifier fusion', *Applied soft computing* **6**(4), 337–347.

Gagné, C., Sebag, M., Schoenauer, M. & Tomassini, M. (2007), Ensemble learning for free with evolutionary algorithms?, *in* 'Proceedings of the 9th annual conference on Genetic and evolutionary computation', ACM, pp. 1782–1789.

Geman, S., Bienenstock, E. & Doursat, R. (1992), 'Neural networks and the bias/variance dilemma', *Neural computation* **4**(1), 1–58.

Geurts, P. (2001), Pattern extraction for time series classification, *in* 'Principles of data mining and knowledge discovery', Springer, pp. 115–127.

Giacinto, G., Roli, F. & Fumera, G. (2000), Design of effective multiple classifier systems by clustering of classifiers, *in* 'Pattern Recognition, 2000. Proceedings. 15th International Conference on', Vol. 2, IEEE, pp. 160–163.

Gill, D. J., Freshman, A., Blender, J. A. & Ravina, B. (2008), 'The Montreal Cognitive Assessment as a screening tool for cognitive impairment in Parkinson's disease', *Movement disorders* **23**(7), 1043–1046.

Goetz, C. G., Tilley, B. C., Shaftman, S. R., Stebbins, G. T., Fahn, S., Martinez-Martin, P., Poewe, W., Sampaio, C., Stern, M. B., Dodel, R. et al. (2008), 'Movement disorder society-sponsored revision of the Unified Parkinson's Disease Rating Scale (MDS-UPDRS): Scale presentation and clinimetric testing results', *Movement Disorders* **23**(15), 2129–2170.

Gu, S., Cheng, R. & Jin, Y. (2015), 'Multi-objective ensemble generation', *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **5**(5), 234–245.

Guyon, I. & Elisseeff, A. (2003), 'An introduction to variable and feature selection', *The Journal of Machine Learning Research* **3**, 1157–1182.

Hand, D. J. (2009), 'Measuring classifier performance: a coherent alternative to the area under the ROC curve', *Machine learning* **77**(1), 103–123.

Hansen, L. K. & Salamon, P. (1990), 'Neural network ensembles', *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **12**(10), 993–1001.

Harik, G. R. (1995), Finding multimodal solutions using restricted tournament selection., *in* 'ICGA', pp. 24–31.

Harvey, D. & Todd, M. D. (2014), 'Automated feature design for numeric sequence classification by genetic programming'.

Haykin, S. (1994), *Neural networks: a comprehensive foundation*, Prentice Hall PTR.

Hebb, D. O. (1949), *The organization of behavior: A neuropsychological theory*, Psychology Press.

Heldman, D. A., Giuffrida, J. P., Chen, R., Payne, M., Mazzella, F., Duker, A. P., Sahay, A., Kim, S. J., Revilla, F. J. & Espay, A. J. (2011), 'The modified bradykinesia rating scale for Parkinson's disease: reliability and comparison with kinematic measures', *Movement Disorders* **26**(10), 1859–1863.

Henderson, L., Kennard, C., Crawford, T., Day, S., Everitt, B., Goodrich, S., Jones, F. & Park, D. (1991), 'Scales for rating motor impairment in Parkinson's disease: studies of reliability and convergent validity.', *Journal of Neurology, Neurosurgery & Psychiatry* **54**(1), 18–24.

Hochreiter, S. & Schmidhuber, J. (1997), 'Long short-term memory', *Neural computation* **9**(8), 1735–1780.

Hoff, J., Wagemans, E., Van Hilten, J. et al. (2001), 'Accelerometric assessment of levodopa-induced dyskinesias in Parkinson's disease', *Movement Disorders* **16**(1), 58–61.

Holladay, K. & Robbins, K. (2007), Evolution of signal processing algorithms using vector based genetic programming, *in* 'Digital Signal Processing, 2007 15th International Conference on', IEEE, pp. 503–506.

Holladay, K., Robbins, K. & Von Ronne, J. (2007), FIFTHTM: A Stack Based GP Language for Vector Processing, *in* 'Genetic Programming', Springer, pp. 102–113.

Holland, J. H. (1973), 'Genetic algorithms and the optimal allocation of trials', *SIAM Journal on Computing* **2**(2), 88–105.

Holland, J. H. (1975), *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.*, U Michigan Press.

Hoops, S., Nazem, S., Siderowf, A., Duda, J., Xie, S., Stern, M. & Weintraub, D. (2009), 'Validity of the MoCA and MMSE in the detection of MCI and dementia in Parkinson disease', *Neurology* **73**(21), 1738–1745.

Hopfield, J. J. (1982), 'Neural networks and physical systems with emergent collective computational abilities', *Proceedings of the national academy of sciences* **79**(8), 2554–2558.

Hughes, A. J., Daniel, S. E., Kilford, L. & Lees, A. J. (1992), 'Accuracy of clinical diagnosis of idiopathic Parkinson's disease: a clinico-pathological study of 100 cases.', *Journal of Neurology, Neurosurgery & Psychiatry* **55**(3), 181–184.

Hunt, K. J., Sbarbaro, D., Żbikowski, R. & Gawthrop, P. J. (1992), 'Neural networks for control systems—a survey', *Automatica* **28**(6), 1083–1112.

Hüsken, M. & Stagge, P. (2003), 'Recurrent neural networks for time series classification', *Neurocomputing* **50**, 223–235.

Iansek, R., Huxham, F. & McGinley, J. (2006), 'The sequence effect and gait festination in Parkinson disease: contributors to freezing of gait?', *Movement disorders* **21**(9), 1419–1424.

Jaeger, H. (2001), 'The "echo state" approach to analysing and training recurrent neural networks-with an erratum note', *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report* **148**, 34.

Jaeger, H. (2002), *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach*, GMD-Forschungszentrum Informationstechnik.

Jakobi, N. (2003), 'Harnessing morphogenesis', *On growth, form and computers* pp. 392–404.

Jankovic, J. (2008), 'ParkinsonâĂŹs disease: clinical features and diagnosis', *Journal of Neurology, Neurosurgery & Psychiatry* **79**(4), 368–376.

Jankovic, J., Rajput, A. H., McDermott, M. P., Perl, D. P. et al. (2000), 'The evolution of diagnosis in early Parkinson disease', *Archives of neurology* **57**(3), 369.

Jenner, P. & Olanow, C. W. (1998), 'Understanding cell death in Parkinson's disease', *Annals of neurology* **44**(3), S72–S84.

Jeong, Y.-S., Jeong, M. K. & Omitaomu, O. A. (2011), 'Weighted dynamic time warping for time series classification', *Pattern Recognition* **44**(9), 2231–2240.

Jiang, F., Berry, H. & Schoenauer, M. (2008), Supervised and evolutionary learning of echo state networks, *in* 'Parallel Problem Solving from Nature–PPSN X', Springer, pp. 215–224.

Jin, Y., Okabe, T. & Sendhoff, B. (2004), Neural network regularization and ensembling using multi-objective evolutionary algorithms, *in* 'Evolutionary Computation, 2004. CEC2004. Congress on', Vol. 1, IEEE, pp. 1–8.

Johnson, K. & Rayens, W. (2007), 'Modern classification methods for drug discovery', *Pharmaceutical Statistics Using SAS: A Practical Guide* p. 7.

Jordan, M. I. (1997), 'Serial order: A parallel distributed processing approach', *Advances in psychology* **121**, 471–495.

Kang, S. Y., Wasaka, T., Shamim, E. A., Auh, S., Ueki, Y., Lopez, G. J., Kida, T., Jin, S.-H., Dang, N. & Hallett, M. (2010), 'Characteristics of the sequence effect in Parkinson's disease', *Movement Disorders* **25**(13), 2148–2155.

Kearns, M. & Valiant, L. (1994), 'Cryptographic limitations on learning boolean formulae and finite automata', *Journal of the ACM (JACM)* **41**(1), 67–95.

Keijsers, N. L., Horstink, M. W. & Gielen, S. C. (2003), 'Automatic assessment of levodopa-induced dyskinesias in daily life by neural networks', *Movement disorders* **18**(1), 70–80.

Kira, K. & Rendell, L. A. (1992), A practical approach to feature selection, *in* 'Proceedings of the ninth international workshop on Machine learning', pp. 249–256.

Kishore, A., Espay, A. J., Marras, C., Al-Khairalla, T., Arenovich, T., Asante, A., Miyasaki, J. & Lang, A. E. (2007), 'Unilateral versus bilateral tasks in early asymmetric Parkinson's disease: Differential effects on bradykinesia', *Movement disorders* **22**(3), 328–333.

Klein, J. & Spector, L. (2007), Unwitting distributed genetic programming via asynchronous javascript and xml, *in* 'Proceedings of the 9th annual conference on Genetic and evolutionary computation', ACM, pp. 1628–1635.

Kohavi, R. & John, G. H. (1997), 'Wrappers for feature subset selection', *Artificial intelligence* **97**(1), 273–324.

Koza, J. R. (1990), *Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems*, Stanford University, Department of Computer Science.

Koza, J. R. (1992), *Genetic Programming: vol. 1, On the programming of computers by means of natural selection*, Vol. 1, MIT press.

Koza, J. R. (1994), *Genetic programming II: automatic discovery of reusable programs*, MIT press.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), Imagenet classification with deep convolutional neural networks, *in* 'Advances in neural information processing systems', pp. 1097–1105.

Krogh, A., Vedelsby, J. et al. (1995), 'Neural network ensembles, cross validation, and active learning', *Advances in neural information processing systems* **7**, 231–238.

Kuan, C.-M. & Liu, T. (1995), 'Forecasting exchange rates using feedforward and recurrent neural networks', *Journal of Applied Econometrics* **10**(4), 347–364.

Kuhn, M. (2015), 'caret v6.0-52 [computer software]'.
URL: *http://archive.ics.uci.edu/ml*

Kuncheva, L. I. (2003), That elusive diversity in classifier ensembles, *in* 'Pattern Recognition and Image Analysis', Springer, pp. 1126–1138.

Kuncheva, L. I., Bezdek, J. C. & Duin, R. P. (2001), 'Decision templates for multiple classifier fusion: an experimental comparison', *Pattern Recognition* **34**(2), 299–314.

Lacy, S. E., Lones, M. A., Smith, S. L., Alty, J. E., Jamieson, D., Possin, K. L. & Schuff, N. (2013), Characterisation of movement disorder in Parkinson's disease using evolutionary algorithms,

in 'Proceeding of the fifteenth annual conference companion on Genetic and evolutionary computation conference companion', ACM, pp. 1479–1486.

Langdon, W. (2005), Pfeiffer–a distributed open-ended evolutionary system, *in* 'AISB', Vol. 5, pp. 7–13.

Langdon, W. B., Barrett, S. & Buxton, B. F. (2002), Combining decision trees and neural networks for drug discovery, *in* 'Genetic Programming', Springer, pp. 60–70.

Langdon, W. B. & Buxton, B. F. (2001*a*), Genetic programming for combining classifiers, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)', Citeseer, pp. 66–73.

Langdon, W. B. & Buxton, B. F. (2001*b*), Genetic programming for improved receiver operating characteristics, *in* 'Multiple Classifier Systems', Springer, pp. 68–77.

Langston, J. W. (1989), 'Current theories on the cause of Parkinson's disease.', *Journal of neurology, neurosurgery, and psychiatry* **52**(Suppl), 13.

Lawrence, S., Giles, C. L., Tsoi, A. C. & Back, A. D. (1997), 'Face recognition: A convolutional neural-network approach', *Neural Networks, IEEE Transactions on* **8**(1), 98–113.

LeWitt, P. A. (2008), 'Levodopa for the treatment of Parkinson's disease', *New England Journal of Medicine* **359**(23), 2468–2476.

Li, J.-P., Balazs, M. E., Parks, G. T. & Clarkson, P. J. (2002), 'A species conserving genetic algorithm for multimodal function optimization', *Evolutionary computation* **10**(3), 207–234.

Lin, X., Yang, Z. & Song, Y. (2009), 'Short-term stock price prediction based on echo state networks', *Expert systems with applications* **36**(3), 7313–7317.

Ling, H., Massey, L. A., Lees, A. J., Brown, P. & Day, B. L. (2012), 'Hypokinesia without decrement distinguishes progressive supranuclear palsy from Parkinson's disease', *Brain* **135**(4), 1141–1153.

Liu, Y. & Yao, X. (1999), 'Ensemble learning via negative correlation', *Neural Networks* **12**(10), 1399–1404.

Liu, Y., Yao, X. & Higuchi, T. (2000), 'Evolutionary ensembles with negative correlation learning', *Evolutionary Computation, IEEE Transactions on* **4**(4), 380–387.

Lobo, J. M., Jiménez-Valverde, A. & Real, R. (2008), 'AUC: a misleading measure of the performance of predictive distribution models', *Global Ecology and Biogeography* **17**(2), 145–151.

Lones, M. A., Smith, S. L., Tyrrell, A. M., Alty, J. E. & Jamieson, D. S. (2013), 'Characterising neurological time series data using biologically motivated networks of coupled discrete maps', *BioSystems* **112**(2), 94–101.

Lones, M. A., Tyrrell, A. M., Stepney, S. & Caves, L. (2011), 'Controlling legged robots with coupled artificial biochemical networks', *Advances in Artificial Life, ECAL* pp. 465–472.

Lones, M. A., Tyrrell, A. M., Stepney, S. & Caves, L. S. (2010), Controlling complex dynamics with artificial biochemical networks, *in* 'Genetic Programming', Vol. 6021 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 159–170.

Lones, M., Smith, S. L., Alty, J. E., Lacy, S. E., Possin, K. L., Jamieson, D., Tyrrell, A. M. et al. (2014), 'Evolving classifiers to recognize the movement characteristics of Parkinson's disease patients', *Evolutionary Computation, IEEE Transactions on* **18**(4), 559–576.

Lones, M., Turner, A. P., Caves, L. S., Stepney, S., Smith, S. L., Tyrrell, A. M. et al. (2014), 'Artificial biochemical networks: Evolving dynamical systems to control dynamical systems', *Evolutionary Computation, IEEE Transactions on* **18**(2), 145–166.

Lukoševičius, M. (2012), A practical guide to applying echo state networks, *in* 'Neural Networks: Tricks of the Trade', Springer, pp. 659–686.

Maass, W., Natschläger, T. & Markram, H. (2002*a*), A model for real-time computation in generic neural microcircuits, *in* 'Advances in neural information processing systems', pp. 213–220.

Maass, W., Natschläger, T. & Markram, H. (2002*b*), 'Real-time computing without stable states: A new framework for neural computation based on perturbations', *Neural computation* **14**(11), 2531–2560.

Maclin, R. & Opitz, D. (2011), 'Popular ensemble methods: An empirical study', *arXiv preprint arXiv:1106.0257* .

Mahfoud, S. W. (1992), Crowding and preselection revisited, *in* 'Parallel Problem Solving From Nature, 2', North-Holland.

Mahfoud, S. W. (1995), A comparison of parallel and sequential niching methods, *in* 'Conference on genetic algorithms', Vol. 136, Citeseer, p. 143.

Manson, A., Brown, P., O'sullivan, J., Asselman, P., Buckwell, D. & Lees, A. (2000), 'An ambulatory dyskinesia monitor', *Journal of Neurology, Neurosurgery & Psychiatry* **68**(2), 196–201.

Martinez-Martin, P., Gil-Nagel, A., Gracia, L. M., Gómez, J. B., Martínez-Sarriés, J. & Bermejo, F. (1994), 'Unified Parkinson's disease rating scale characteristics and structure', *Movement disorders* **9**(1), 76–83.

McCulloch, W. S. & Pitts, W. (1943), 'A logical calculus of the ideas immanent in nervous activity', *The Bulletin of Mathematical Biophysics* **5**(4), 115–133.

McGill, R., Tukey, J. W. & Larsen, W. A. (1978), 'Variations of box plots', *The American Statistician* **32**(1), 12–16.

McKay, R. & Abbass, H. (2001*a*), 'Anti-correlation: A diversity promoting mechanisms in ensemble learning', *The Australian Journal of Intelligent Information Processing Systems* **3**(4), 139–149.

McKay, R. & Abbass, H. (2001*b*), Anticorrelation measures in genetic programming, *in* 'Australasia-Japan Workshop on Intelligent and Evolutionary Systems', pp. 45–51.

Mengshoel, O. J. & Goldberg, D. E. (1999), Probabilistic crowding: Deterministic crowding with probabilistic replacement, *in* 'Proc. of the Genetic and Evolutionary Computation Conference (GECCO-99)', pp. 409–416.

Miller, J. F. & Thomson, P. (2000), Cartesian genetic programming, *in* 'Genetic Programming', Springer, pp. 121–132.

Minsky, M. & Papert, S. (1969), 'Perceptrons'.

Nasreddine, Z. S., Phillips, N. A., Bédirian, V., Charbonneau, S., Whitehead, V., Collin, I., Cummings, J. L. & Chertkow, H. (2005), 'The Montreal Cognitive Assessment, MoCA: a brief screening tool for mild cognitive impairment', *Journal of the American Geriatrics Society* **53**(4), 695–699.

Nicolau, M., Schoenauer, M. & Banzhaf, W. (2010), Evolving genes to balance a pole, *in* 'Genetic Programming', Springer, pp. 196–207.

Nordqvist, C. (2009), 'What is Alzheimer's Disease? What Causes Alzheimer's Disease?', http://www.medicalnewstoday.com/articles/159442.php.

Oliveira, L. S., Morita, M. & Sabourin, R. (2006), Feature selection for ensembles using the multi-objective optimization approach, *in* 'Multi-Objective Machine Learning', Springer, pp. 49–74.

Opitz, D. W. (1999), Feature selection for ensembles, *in* 'AAAI/IAAI', pp. 379–384.

Osterrieth, P. A. (1944), 'Le test de copie d'une figure complexe; contribution à l'étude de la perception et de la mémoire.', *Archives de psychologie* .

Parkinson, J. (1817), *An essay on the shaking palsy*, Printed by Whittingham and Rowland for Sherwood, Neely, and Jones.

Patel, S., Chen, B.-r., Buckley, T., Rednic, R., McClure, D., Tarsy, D., Shih, L., Dy, J., Welsh, M. & Bonato, P. (2010), Home monitoring of patients with Parkinson's disease via wearable technology and a web-based application, *in* 'Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE', IEEE, pp. 4411–4414.

Patel, S., Lorincz, K., Hughes, R., Huggins, N., Growdon, J., Standaert, D., Akay, M., Dy, J., Welsh, M. & Bonato, P. (2009), 'Monitoring motor fluctuations in patients with Parkinson's disease using wearable sensors', *Information Technology in Biomedicine, IEEE Transactions on* **13**(6), 864–873.

Pepe, M. S. (2000), 'Receiver operating characteristic methodology', *Journal of the American Statistical Association* **95**(449), 308–311.

Perkis, T. (1994), Stack-based genetic programming, *in* 'Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on', IEEE, pp. 148–153.

Poewe, W., Antonini, A., Zijlmans, J. C., Burkhard, P. R. & Vingerhoets, F. (2010), 'Levodopa in the treatment of Parkinson's disease: an old drug still going strong', *Clinical interventions in aging* **5**, 229.

Poli, R., Page, J. & Langdon, W. B. (1999), Smooth uniform crossover, sub-machine code gp and demes: A recipe for solving high-order boolean parity problems, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference', Vol. 2, pp. 1162–1169.

Poli, R. et al. (1997), Evolution of graph-like programs with parallel distributed genetic programming., *in* 'ICGA', Citeseer, pp. 346–353.

Provost, F. J., Fawcett, T. & Kohavi, R. (1998), The case against accuracy estimation for comparing induction algorithms., *in* 'ICML', Vol. 98, pp. 445–453.

Quick, T., Nehaniv, C. L., Dautenhahn, K. & Roberts, G. (2003), Evolving embodied genetic regulatory network-driven control systems, *in* 'Advances in Artificial Life', Springer, pp. 266–277.

Quinlan, J. R. (1993), 'C4.5: Programs for machine learning'.

Quinlan, R. (2004), 'Data mining tools see5 and c5.0'.

Rajput, A. H., Fenton, M. E., Birdi, S., Macaulay, R., George, D., Rozdilsky, B., Ang, L. C., Senthilselvan, A. & Hornykiewicz, O. (2002), 'Clinical–pathological study of levodopa complications', *Movement disorders* **17**(2), 289–296.

Rajput, A., Rozdilsky, B. & Rajput, A. (1991), 'Accuracy of clinical diagnosis in Parkinsonism–a prospective study.', *The Canadian journal of neurological sciences. Le journal canadien des sciences neurologiques* **18**(3), 275–278.

Rascol, O., Brooks, D. J., Korczyn, A. D., De Deyn, P. P., Clarke, C. E. & Lang, A. E. (2000), 'A five-year study of the incidence of dyskinesia in patients with early Parkinson's disease who were treated with ropinirole or levodopa', *New England Journal of Medicine* **342**(20), 1484–1491.

Ratanamahatana, C. A. & Keogh, E. (2004), Making time-series classification more accurate using learned constraints, *in* 'Proceedings of the Fourth SIAM International Conference on Data Mining', SIAM, p. 11.

Rey, A. (1941), 'L'examen psychologique dans les cas d'encéphalopathie traumatique.(les problems.).', *Archives de psychologie* .

Rodrigues, J. P., Mastaglia, F. L. & Thickbroom, G. W. (2009), 'Rapid slowing of maximal finger movement rate: fatigue of central motor control?', *Experimental brain research* **196**(4), 557–563.

Rodríguez, J. J. & Alonso, C. J. (2004), Interval and dynamic time warping-based decision trees, *in* 'Proceedings of the 2004 ACM symposium on Applied computing', ACM, pp. 548–552.

Rosenblatt, F. (1958), 'The perceptron: a probabilistic model for information storage and organization in the brain.', *Psychological review* **65**(6), 386.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1988), 'Learning representations by back-propagating errors', *Cognitive modeling* **5**.

Ruta, D. & Gabrys, B. (2005), 'Classifier selection for majority voting', *Information fusion* **6**(1), 63–81.

Salmen, M. & Ploger, P. G. (2005), Echo state networks used for motor control, *in* 'Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on', IEEE, pp. 1953–1958.

Schrag, A. & Quinn, N. (2000), 'Dyskinesias and motor fluctuations in Parkinson's disease a community-based study', *Brain* **123**(11), 2297–2305.

Scott, M., Niranjan, M. & Prager, R. (1998), Realisable classifiers: Improving operating performance on variable cost problems, *in* 'Proceedings of the British Machine Conference, pages', pp. 31–1.

Selkoe, D. & Lansbury Jr, P. (1999), Alzheimer's disease is the most common neurodegenerative disorder, *in* G. J. Siegel, B. W. Agranoff, R. W. Albers, S. K. Fisher, M. D. Uhler et al., eds, 'Basic neurochemistry: molecular, cellular and medical aspects, 6th edn.', Lippincott-Raven, Philadelphia.

Sharman, K., Alcázar, A. I. E. & Li, Y. (1995), Evolving signal processing algorithms by genetic programming, *in* 'Genetic Algorithms in Engineering Systems: Innovations and Applications, 1995. GALESIA. First International Conference on (Conf. Publ. No. 414)', IET, pp. 473–480.

Siedlecki, W. & Sklansky, J. (1989), 'A note on genetic algorithms for large-scale feature selection', *Pattern recognition letters* **10**(5), 335–347.

Simard, P. Y., Steinkraus, D. & Platt, J. C. (2003), Best practices for convolutional neural networks applied to visual document analysis, *in* 'null', IEEE, p. 958.

Singh, G. & Deb, K. (2006), Comparison of multi-modal optimization algorithms based on evolutionary algorithms, *in* 'Proceedings of the 8th annual conference on Genetic and evolutionary computation', ACM, pp. 1305–1312.

Sirlantzis, K., Fairhurst, M. C. & Guest, R. M. (2002), An evolutionary algorithm for classifier and combination rule selection in multiple classifier systems, *in* 'Pattern Recognition, 2002. Proceedings. 16th International Conference on', Vol. 2, IEEE, pp. 771–774.

Skowronski, M. D. & Harris, J. G. (2006), Minimum mean squared error time series classification using an echo state network prediction model, *in* 'Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on', IEEE, pp. 4–pp.

Skowronski, M. D. & Harris, J. G. (2007), 'Automatic speech recognition using a predictive echo state network classifier', *Neural networks* **20**(3), 414–423.

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

Stepney, S. (2012), Nonclassical computationâĂŤa dynamical systems perspective, *in* 'Handbook of natural computing', Springer, pp. 1979–2025.

Teller, A. & Veloso, M. (1995), 'Program evolution for data mining', *International Journal of Expert Systems Research and Applications* **8**, 213–236.

Teller, A. & Veloso, M. (1997), 'PADO: A new learning architecture for object recognition', *Symbolic visual learning* pp. 77–112.

Thanvi, B., Lo, N. & Robinson, T. (2007), 'Levodopa-induced dyskinesia in Parkinson's disease: clinical features, pathogenesis, prevention and treatment', *Postgraduate medical journal* **83**(980), 384–388.

Tong, M. H., Bickett, A. D., Christiansen, E. M. & Cottrell, G. W. (2007), 'Learning grammatical structure with echo state networks', *Neural Networks* **20**(3), 424–432.

Tsanas, A., Little, M. A., McSharry, P. E., Spielman, J. & Ramig, L. O. (2012), 'Novel speech signal processing algorithms for high-accuracy classification of Parkinson's disease', *Biomedical Engineering, IEEE Transactions on* **59**(5), 1264–1271.

Turing, A. (1948), 'Intelligent machinery. Report for National Physical Laboratory. Reprinted in Ince, DC (editor). 1992. Mechanical Intelligence: Collected works of AM Turing'.

Valiant, L. G. (1984), 'A theory of the learnable', *Communications of the ACM* **27**(11), 1134–1142.

Verplancke, T., Van Looy, S., Steurbaut, K., Benoit, D., De Turck, F., De Moor, G. & Decruyenaere, J. (2010), 'A novel time series analysis approach for prediction of dialysis in critically ill patients using echo-state networks', *BMC medical informatics and decision making* **10**(1), 4.

Verstraeten, D., Schrauwen, B., Stroobandt, D. & Van Campenhout, J. (2005), 'Isolated word recognition with the liquid state machine: a case study', *Information Processing Letters* **95**(6), 521–528.

Website of the Parkinson's Disease Society (2013), Available at http://www.parkinsons.org.uk.

Widrow, B., Hoff, M. E. et al. (1960), 'Adaptive switching circuits.'.

Williams, R. J. & Zipser, D. (1989), 'A learning algorithm for continually running fully recurrent neural networks', *Neural computation* **1**(2), 270–280.

Wolpert, D. H. & Macready, W. G. (1997), 'No free lunch theorems for optimization', *Evolutionary Computation, IEEE Transactions on* **1**(1), 67–82.

Xi, X., Keogh, E., Shelton, C., Wei, L. & Ratanamahatana, C. A. (2006), Fast time series classification using numerosity reduction, *in* 'Proceedings of the 23rd international conference on Machine learning', ACM, pp. 1033–1040.

Xie, F., Song, A. & Ciesielski, V. (2012), Event detection in time series by genetic programming, *in* 'Evolutionary Computation (CEC), 2012 IEEE Congress on', IEEE, pp. 1–8.

Xing, Z., Pei, J. & Keogh, E. (2010), 'A brief survey on sequence classification', *ACM SIGKDD Explorations Newsletter* **12**(1), 40–48.

Yamazaki, T. & Tanaka, S. (2007), 'The cerebellum as a liquid state machine', *Neural Networks* **20**(3), 290–297.

Yang, J. & Honavar, V. (1998), Feature subset selection using a genetic algorithm, *in* 'Feature extraction, construction and selection', Springer, pp. 117–136.

Yao, X. (1993), 'A review of evolutionary artificial neural networks', *International journal of intelligent systems* **8**(4), 539–567.

Yao, X. (1999), 'Evolving artificial neural networks', *Proceedings of the IEEE* **87**(9), 1423–1447.

Yao, X. & Liu, Y. (1997), 'A new evolutionary system for evolving artificial neural networks', *Neural Networks, IEEE Transactions on* **8**(3), 694–713.

Yokoe, M., Okuno, R., Hamasaki, T., Kurachi, Y., Akazawa, K. & Sakoda, S. (2009), 'Opening velocity, a novel parameter, for finger tapping test in patients with Parkinson's disease', *Parkinsonism & Related Disorders* **15**(6), 440–444.

Zhou, Z.-H., Wu, J. & Tang, W. (2002), 'Ensembling neural networks: many could be better than all', *Artificial intelligence* **137**(1), 239–263.

Zitzler, E., Laumanns, M., Thiele, L., Zitzler, E., Zitzler, E., Thiele, L. & Thiele, L. (2001), 'SPEA2: Improving the strength pareto evolutionary algorithm'.

Zweig, M. H. & Campbell, G. (1993), 'Receiver-operating characteristic (ROC) plots: a fundamental evaluation tool in clinical medicine.', *Clinical chemistry* **39**(4), 561–577.