

Thesis Project Proposal

Feature Selection for Brain Computer Interface using Genetic Algorithm

Timothy Mahoney (42360366)

Supervised by: Assoc. Prof. Marcus Gallagher

November 5th, 2018

letter

acknowledgments

Marcus

Brendan and Nick

Abstract

# Table of Contents

[Table of Contents iv](#_Toc528170038)

[1 Introduction 1](#_Toc528170039)

[1.1 Aims and Scope 1](#_Toc528170040)

[2 Background 3](#_Toc528170041)

[2.1 Electroencephalography 3](#_Toc528170042)

[2.1.1 Cortical Homunculus 4](#_Toc528170043)

[2.1.2 Frequency Bands 4](#_Toc528170044)

[2.1.3 Event Related De/Synchronisation 5](#_Toc528170045)

[2.1.4 Artefacts 5](#_Toc528170046)

[2.2 Brain Computer Interface 6](#_Toc528170047)

[2.2.1 Feature Extraction 7](#_Toc528170048)

[2.3 Feature Selection 8](#_Toc528170049)

[2.4 Computational Optimisation 8](#_Toc528170050)

[2.4.1 Genetic Algorithm 9](#_Toc528170051)

[2.5 Classification Algorithms 10](#_Toc528170052)

[2.5.1 Artificial Neural Network 10](#_Toc528170053)

[2.5.2 Support Vector Machine 10](#_Toc528170054)

[2.6 Previous Related Study 11](#_Toc528170055)

[3 Methods 12](#_Toc528170056)

[3.1 Initial Filtering 12](#_Toc528170057)

[3.2 Automated Artefact Removal 12](#_Toc528170058)

[3.2.1 Independent Component Analysis 13](#_Toc528170059)

[3.2.2 Component Feature Extraction 14](#_Toc528170060)

[3.2.3 Artificial Neural Network 15](#_Toc528170061)

[3.2.4 Channel Reconstruction 16](#_Toc528170062)

[3.3 Surface Laplacian 17](#_Toc528170063)

[3.4 Feature Extraction 17](#_Toc528170064)

[3.5 Feature Selection 18](#_Toc528170065)

[3.5.1 Genetic Algorithm Implementation 18](#_Toc528170066)

[3.5.2 Filter Implementation 20](#_Toc528170067)

[4 Risk analysis 25](#_Toc528170068)

[4.1 Breach of Copyright 25](#_Toc528170069)

[4.2 Data Loss 26](#_Toc528170070)

[4.3 Underestimation of Time Requirement 26](#_Toc528170071)

[5 Project Plan 27](#_Toc528170072)

[6 Conclusion 29](#_Toc528170073)

[1 Bibliography 30](#_Toc528170074)

List of figures

List of tables

List of Acronyms

BCI

EEG

EOG

EMG

ECG

SVM

ANN artificial neural net

Artefact

Motor Movement (MM)

Motor Imagery (MI)

ERD ERS

ICA

AAR

SL

GA PGA

GBE

Preface

The report clearly defines and explains the background of a Master of Engineering Science thesis project to be conducted by the author at the University of Queensland. A thorough exploration of background material is covered in Section 2, followed by an investigation into previous similar studies in Section 3. A risk analysis is displayed in Section 4 and Section 5 covers the project plan, including milestones, assessment dates and a timeline.

# Introduction

Advances in computational power and machine learning techniques has enthused global interest in the development Brain Computer Interfacing (BCI) technology. With applications ranging from motor and sensory prosthetics, to communication and even connection to the internet of things, BCI advancement has the potential to provide significant benefits to society. Recently, Facebook has begun development of a BCI to enable typing via thought [1], engineers at Johns Hopkins University have created a BCI capable of controlling an upper limb prosthesis [2], even Elon Musk’s latest venture is in BCI technology with his startup company *Neuralink* [3].

Currently however, effectively classifying neural data in an online and non-invasive scenario is no trivial problem. Features relevant to neural events include temporal patterns in a variety of frequency bandwidths across many spatially separated electrodes. Due to the high spatial resolution and frequency bandwidth of non-invasive electroencephalographic (EEG) signals, one hurdle slowing BCI development is determining what data is useful and what can be ignored. Selecting an appropriate subset of features from such a large set of candidates could reduce the computational complexity and hardware requirements of such a system, thereby improving the systems feasibility.

## Aims and Scope

This project examines various methods of feature selection to produce a heuristically optimal subset of features using the natural selection based optimization technique Genetic Algorithm (GA), for the purpose of motor-imagery event classification, from non-invasive neural data (EEG). The specific aims can be summarised in three points:

1. Use Genetic Algorithm to establish a heuristically optimal subset of features from the full EEG feature set.
2. The subset should both significantly reduce the number of features *and* result in low classification error
3. The data sampling and processing methods should simulate an online BCI system such that it should be: automated, computationally efficient and robust to slow sampling.

Specifically, after filtering and artefact removal, we extract features consisting of temporal data spanning standard EEG frequency bands from 64-channel EEG recordings. The extracted features are passed to the Genetic Algorithm selector which is implemented as a wrapper, such that the metric used to evaluate a feature set’s fitness is classification error with a Support Vector Machine (SVM).

DIAGRAM: 64 CHANNEL DATA > FILTERING and ARTEFACT REMOVAL > FEATURE EXTRACTION > GA <> SVM

The dataset being used was created in 2004 by the BCI2000 instrumentation system and contributed to the PhysioNet database [4] [5]. The dataset contains over 1500 one- and two-minute EEG recordings from 109 participants. The system used 64 channel EEG with a sampling rate of 160 Hz. It consisted of 14 experimental runs including two baseline recordings (eyes open, then closed) and three repeats of four tasks. The datasets being considered for this project are of Task 2: the subject *imagines* closing their left or right fists (motor imagery, MI) for a period in response to visual cues. We pose the task as a binary classification problem, with labels corresponding to left (L) or right (R) MI events. The aim of the BCI for this project is to classify these events using the EEG recordings.

MATLAB will be the primary software package being used to import, sample, optimise and classify the data. This is because the author is familiar with the software package and there is a substantial pool of supporting documentation available for correctly implementing most, if not all, of the processes involved with this project.

It should be noted that there are assumptions being made in this project. Previous studies have investigated the effects of user specific parameter selection under the motivation that EEG responses can be dependent on the subject [6]. Due to the nature of the dataset, implementing the BCI in this way may produce less meaningful results as the number of experimental runs per person is too small. Lastly, the focus topic for this project is the optimisation methods being implemented, hence the classification algorithms are simply used to evaluate the performance of the features selected and will not be investigated in detail.

# Background

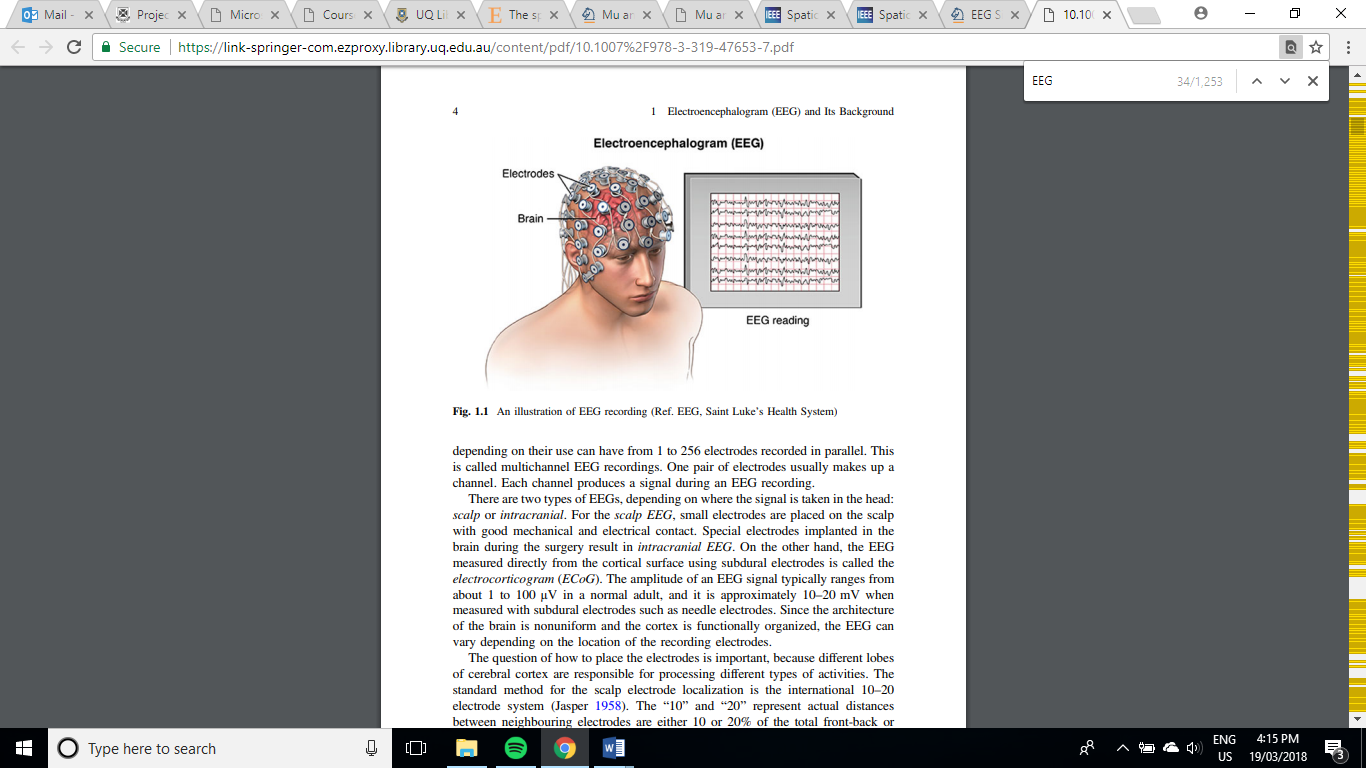
BCI systems are the culmination of knowledge across fields spanning biology, neuroscience, electrical and biomedical signal processing, and machine learning to name a few. The following section covers, in detail, the large amount of background knowledge required for this project due to the multidisciplinary nature of BCI technology. All background information is referenced accordingly should the reader be interested in further reading.

## Electroencephalography

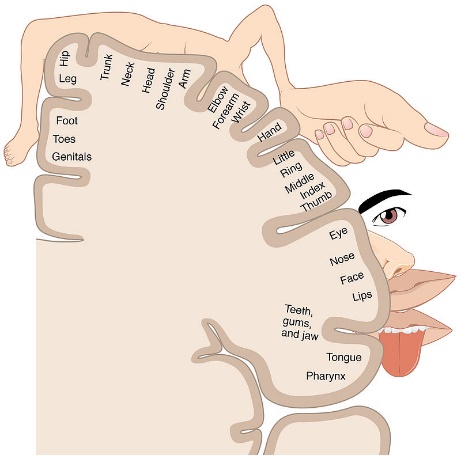
Electroencephalography (EEG) is the recording of action potentials originating from the brain. It is most often used in the study of brain functionality or the diagnoses of neurological disorders [7]. EEG signals are sampled using several small electrodes placed on the scalp, which can detect minute potential differences non-invasively on the surface of the skin [7]. Through proper filtering and signal processing techniques, the summation of electrical activity from local neurons can be isolated and displayed allowing professionals to examine neural activity pertaining to each electrode [7].

The exact placement of the electrodes is a generally subject to the standardised International 10-20 System, whereby the distance between left-right and front-back electrodes is either 10% or 20% of the total skull distance [8]. The electrodes are labelled according to the area of the brain, or lobe, that they’re reading from. These are pre-frontal (Fp), frontal (F), temporal (T), parietal (P), occipital (O) and central (C) as seen in Figure 1 [7] [8]. The number of electrodes in total varies, systems can sample anywhere from 1 to 256 EEG signals in parallel [7].

Figure 1 (left) Standard EEG labelling of each lobe. (right) Examples of EEG positioning and recordings [7]



## Cortical Homunculus

Neural sensory and motor function are generally thought to be confined to certain spatial locations within the brain. A cortical homunculus is an illustration describing this spatial separation as observed from the brains surface along a coronal cross section at either the sensory or motor cortex. Considering the homunculus depicted in Figure X, we can see the tissue associated with hand control is a relatively large portion approximately half way down the cortex. While we do aim to find an appropriate and small feature set using primarily GA, for the purposes of this project we can perform some preliminary reduction of the feature space by neglecting the peripheral channels of the EEG as we can, with confidence, say that these channels are unlikely to relate to MI events associated with hand movement.

## 2.1.2 Frequency Bands

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Band | Delta | Theta | Mu | Alpha | Beta | Gamma |
| Frequency (Hz) | 0.5 - 4 | 4 – 7 | 7-12 | 11-15 | 15-31 | >31 |

The frequency components of EEG are considered some of the most important criteria for interpreting brain function [7]. The bandwidth of these components is commonly defined in Table X.

The presence or absence of each component is generally associated with a particular type of brain function or state [7]. For instance, the three bands which are commonly associated with sensory-motor function are alpha, beta and mu waves, and are hence the three frequency bands being used for feature extraction in this project.

## 2.1.3 Event Related De/Synchronisation

There is evidence to suggest that in addition to the amplitude of each bandwidth indicating a neurological event, temporal patterns referred to as event related de/synchronisation (ERD, ERS) between particular bands may correlate to the mind processing information [9] [10]. ERS occurs when there is an observable increase in neural activity due to the synchronisation of neurons firing action potentials at a given frequency band. Likewise, ERD is the phenomena of de-synchronised neurons cancelling each other out, thereby reducing the observable electrical potential of a given frequency band. ERS and ERD events shown to relate to sensory-motor activity last over a period of seconds. In order to consider this information in the feature set for this project, we include features extracted from previous samples as inputs into our classifier.

## 2.1.4 Artefacts

Signals included in EEG recordings that do not originate from neural potentials are referred to as artefacts. They can be introduced from other physiological events such electromyographic signals (EMG) from muscles, electrooculographic signals (EOG) from eye movements or even electrocardiographic signals (ECG) from the heart. They can also exist as noise from poor electrode contact or 50/60Hz mains coupling. EEG signals can easily be corrupted by artefacts as their amplitude is in the range of 10-100μV, hence filtering techniques have been thoroughly explored in previous studies to establish clean signals for professional examination or further processing [7].

Three commonly used artefact removal methods include:

1. Epoch removal – simply manually remove any epoch samples corrupted with artefacts
2. Manual artefact removal – manually remove components likely to be artefacts before reconstructing the signals
3. Automated artefact removal (AAR) – implement automated classification and removal of artefacts

As this project aims to produce a solution which is capable of online classification, implementing AAR is the ideal method of artefact removal. Artefacts are commonly classified by first separating the n-channel EEG into some k components (k≤n), and then considering features of these components such as their temporal pattern, frequency response and spatial distribution. Components are isolated using Independent Component Analysis (ICA).

ICA is often likened to isolating voices in a crowd. Given a number of spatially separated microphones, ICA is able to isolate specific voices (or components) by combining information from each. In the case of EEG data, we have 64 channels all sampling a symphony of neural information and artefacts. Components isolated from this data may be attributed to either one of these groups, if we can identify which we can remove artefact components before reconstructing the channels. The automated ICA implementation for this project will be discussed in more detail in the *Methods* section.

## Brain Computer Interface

A brain-computer interface (BCI) aims to establish a pathway between neuropsychological signals and an external device [11]. One potential goal of such an interface is to provide a substitute method of communicating or controlling extremities for people with neuromuscular impairments [7] [11]. The ability to classify mental imagination events would allow patients to regain some degree of freedom whether it be through wheelchair or neuroprosthesis control, or communication. With proper training a person could repurpose their neural pathways for any number of applications to improve quality of life. However, for such BCI systems to be considered a useful alternative control method they need to achieve critical performance requirements for robustness, consistency and safety [11].

Mason and Birch described a general framework for BCI systems as a way of standardising ideas and terms for these devices to aid with their development and comparison [12]. The framework consists of 6 steps [12] [7]:

**Brain activity measurement** - the sampling of the raw EEG data

**Pre-processing** – digital to analogue conversion, initial filtering and artefact removal

**Feature extraction** – retrieve meaningful features of the EEG data which can discriminate between mental states being used to control the system

**Classification** – employ some algorithm which can classify mental states based on the extracted features.

**Translation into a command** – relate the result of classification with a response in the target device.

**Feedback** – sensory feedback to the user not only allows them to know that the device is working, but also helps train their brain to use it.

The proposed project will focus on the area of feature extraction but will include elements of pre-processing and classification methods.

Commonly used mental responses for controlling a BCI are motor imagery (MI), where the subject imagines moving without performing the movement, and motor movement (MM), which is the brains response to an actual motor movement being made. Of these two tasks, this project will only consider MI as this medium is inherently more suited to applications such as limb replacement.

## Feature Extraction

One of the most critical elements of a functional BCI is effective feature extraction. As discussed in sections 2.1.1, 2.1.2 and 2.1.3, the features being extracted span frequency, space and time in order to best utilise known phenomena observable in the EEG data relating to MI events. Prior to feature extraction however, the data undergoes only one more process known as Surface Laplacian (SL).

SL is a transformation commonly used to improve spatial resolution of EEG data [13]. When EEG data is sampled, it is measured as a potential difference between each channel and some reference electrode. This means there is often redundant information common to neighbouring electrodes due to their close proximity. SL is a matrix multiplication which effectively decouples the channels from the reference electrode, transforming the recordings into comparative measurements between neighbours. There are several SL techniques, each with varying complexity as described by Carvalhaes and de Barros [13], however the method used in this project only considers the simple Finite Difference Method as described in the reference and presented in Appendix A. This method makes the assumptions that the scalp surface is locally flat and that electrodes are equidistant [13]. These assumptions are considered reasonable for the purposes of this Thesis. SL is the final step of feature extraction before employing feature selection.

## Feature Selection

Consider the case where a 128 channel EEG was divided into its 5 frequency bands and the mean of each band for the previous 5 samples was taken as the feature set, resulting in a 3200 long feature set. A feature vector this large introduces several problems in the classification stage, such as [14]:

1. Over complicating the classification model
2. Long training time
3. Slow real time BCI performance
4. Increased risk of overfitting
5. Curse of dimensionality – small ratio of data to number of features

Feature selection attempts to counter these problems by selecting a subset of features without incurring much loss of information [14]. The selection process follows the premise that certain features may correlate with each other (termed: *redundancy*), whereas others may have little to do with classification so including them within the feature vector results in no information gain (*irrelevance*). Feature selection algorithms are essentially a search problem across candidate feature subsets, for some set which is at least heuristically optimal concerning some evaluation score. Score metrics depend on the specific algorithm being applied, the three types are: filter, wrapper and embedded [15]. Filter algorithms select the feature subset independent of the prediction method with metrics such as correlation [16], whereas wrappers assess the subset using the predictor and embedded methods optimise the subset in the process of training the predictor [15]. This project initially considers the use of a filter feature selector before expanding into methods involving wrappers.

## Computational Optimisation

Computational optimisation uses algorithms to minimise or maximise an objective function. In the case of EEG feature selection, the objective function would be the metrics we chose to optimise for (e.g. minimise redundancy, maximise relevance) over the space of feature subsets. If the space of subsets includes every combination of the feature vector proposed earlier (3200 features), there exists over 10963 combinations of features. Brute force search methods over this space are not computationally feasible and as such optimisation problems involving large spaces over many dimensions have led to the creation of novel search algorithms to find *good enough* optimal solutions, these are referred to as *heuristic* algorithms. These methods aim to computationally balance *exploration* and *exploitation* of the search space. Exploration is the wide spread search of the space, often at random, in order to avoid focussing on what could only be local optima, whereas exploitation attempts to converge on the ideal local solution based on information the algorithm has already discovered. This project implements one such heuristic search method base on the evolutionary phenomena: natural selection. The method is Genetic Algorithm (GA).

## Genetic Algorithm

As is the case with many optimisation methods, GA is an algorithm designed to mimic the principals of a naturally occurring system. The method takes the process of evolution, such that the genetic code of the parents is inherited by the child, and simulates a ‘survival of the fittest’ scenario over a series of generations [16]. With respect to this project, the genetic code of the parents/child would be a combination of EEG features and the measure of ‘fitness’ could be the metrics of redundancy and relevance for that particular combination. The method starts with randomly generated parents in the search space. The next generation is computed with a combination of three *genetic operators* [16]:

**Reproduction operator:** poor solutions are neglected, whereas fit solutions are passed down. The number of candidates is preserved by including more children with strong performance to replace the those lost. The resulting candidates are now part of what is referred to as the *mating pool*. It can be seen that the reproduction operator does not result in any new solutions, they are formed in the remaining two operators. Passing on the best candidates to the next generation directly is a method of preserving the best solution, these best candidates are referred to as *elite children*.

**Crossover Operator:** there are a few variations of this operator, but generally two candidates (the parents) from the mating pool are chosen randomly and a subsection of their genetic structure is switched to form children. This operation supplies new candidates with genes that have been previously shown to perform well in the reproduction stage, in the hope that they may perform better than their parents.

**Mutation Operator:** the remaining codes are now subject to mutation, where there exists a small probability that one or more of their genes may be altered. This introduces exploration into the algorithm, allowing it to traverse previously unseen areas of the search space in pursuit of better optima.

The scores of each candidate are computed after these operations in order to inform the algorithm how best to manipulate the next generation. Bit strings pertaining to high scores are carried through to next generations, whereas poor solutions are not. New generations are computed continuously until some stopping criteria is met, for example a certain number of generations or a high score threshold. The process is described quite well by Deb, K. who uses the example of minimising materials used to produce a cylinder of set volume as a function of diameter and height [16]. It is worth noting that this algorithm in no way encourages a significant reduction in the number of features selected, only that a high scoring solution be found. Creatively altering or adding to the method is required to achieve this, which is further discussed in the *Methods* section.

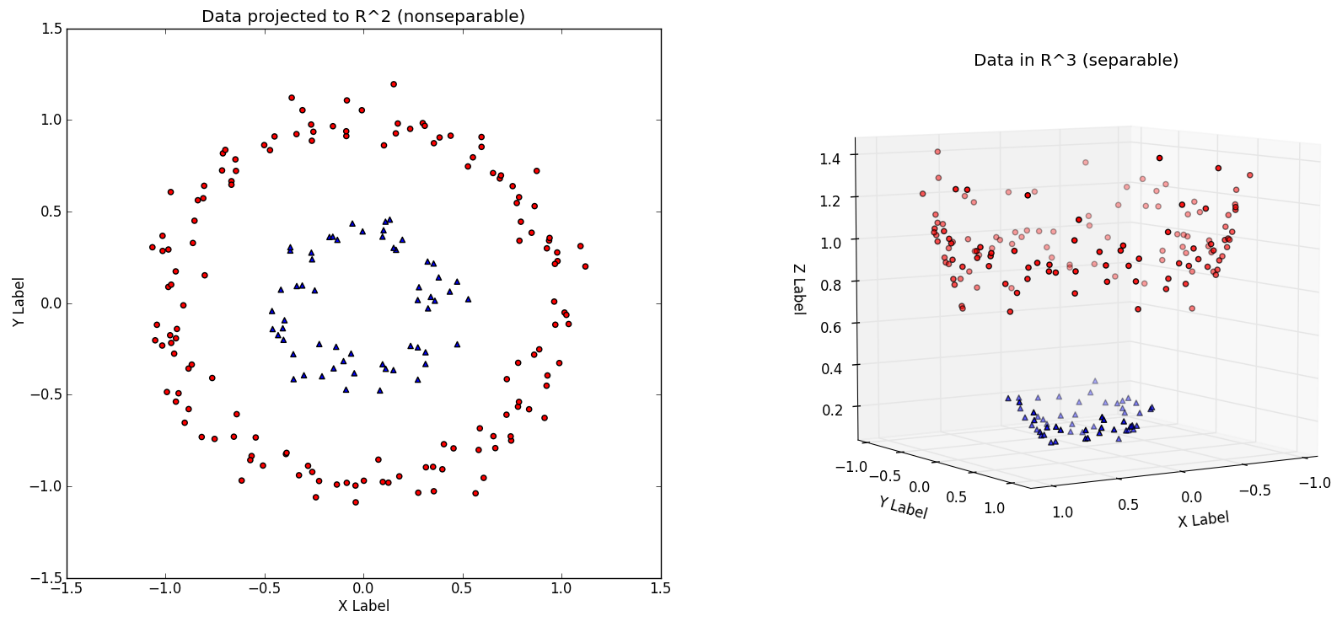
## Classification Algorithms

Classifiers are used in this project for AAR and MI event classification. The scope largely consists of investigation into optimisation techniques however, as such exploring and tuning of machine learning aspects of the project are relatively brief. The classifiers used in the project include an Artificial Neural Network (ANN) and a Support Vector Machine (SVM). These methods were chosen as they have been used previously with EEG classification [17] [18] [19] and MATLAB documentation on their implementation is extensive.

## Artificial Neural Network

An ANN mimics the principals of neural pathways in the brain. A single neuron takes weighted inputs from one or more neurons in a previous layer, where it can then signal other neurons in subsequent layers. The weights between the neurons are updated as learning progresses, tuning the network to accept data features as inputs and predict its class based off supervised learning from a training set. A wide variety of network topologies exist, however this project will only be considering simple feedforward networks with a small number of layers. Hyperparameters consist of the number of layers, the number of neurons in each layer, the training function, the performance metric and the stopping criteria.

## Support Vector Machine

Support Vector Machines are simple discriminant classifiers which construct an optimal hyperplane in the sample space between classes from a training set. The resulting hyperplane is then the discriminant used to classify new data points. Usually, the best hyperplane is taken as the one that maximises the separation of the classes. In cases where the data is not linearly separable, SVM can still work quite well if the data is extended to a higher number of dimensions via some kernel function. Kernel functions are used to map the features to some higher dimension where they are seen as separable and hence can be classified by the SVM, an example of this is shown in Figure X.

SVM parameters include the kernel function scale (gamma) and the soft margin penalty scalar (*C*). Variations in gamma relate to the gradient of the kernel function, if the data for a given class is sparse then gamma should be small to reduce the slope of the kernel function describing the data. The soft margin scalar *C* relates to how much the classifier penalises data found outside the soft margin for its class, a large *C* results in a more strictly trained SVM.

## Previous Related Study

*EEG Classification with Advanced Feature Extraction and Machine Learning* by Alomari et al. uses the same dataset proposed for use with this project [20] [4] [5]. They propose a system which focuses on feature extraction, rather than selection. The features include Energy, Power and Mean metrics for ERD and ERS events of the beta and alpha bands, as well as Movement Related Cortical Potentials (MRCP. The feature set is evaluated using an ANN and SVM. The results show high classification accuracy: 97.1% using SVM and 86.5 using ANN. This paper sets a benchmark for this project to aim for and may provide insight into the importance of feature extraction vs. feature selection. Comparison may also highlight limitations in the approaches of both papers despite the same dataset.

# Methods

The following section contains the methods used and decisions made during the project period. Any results displayed in this section are done so to validate the project before moving to subsequent methods, or to justify an alteration in the approach. Results pertaining to the feature selection methods employed are held out and displayed in the *Results* section. The data is segmented into two second epochs sampled at 2Hz for the entirety of the process in order to simulate an online BCI system.

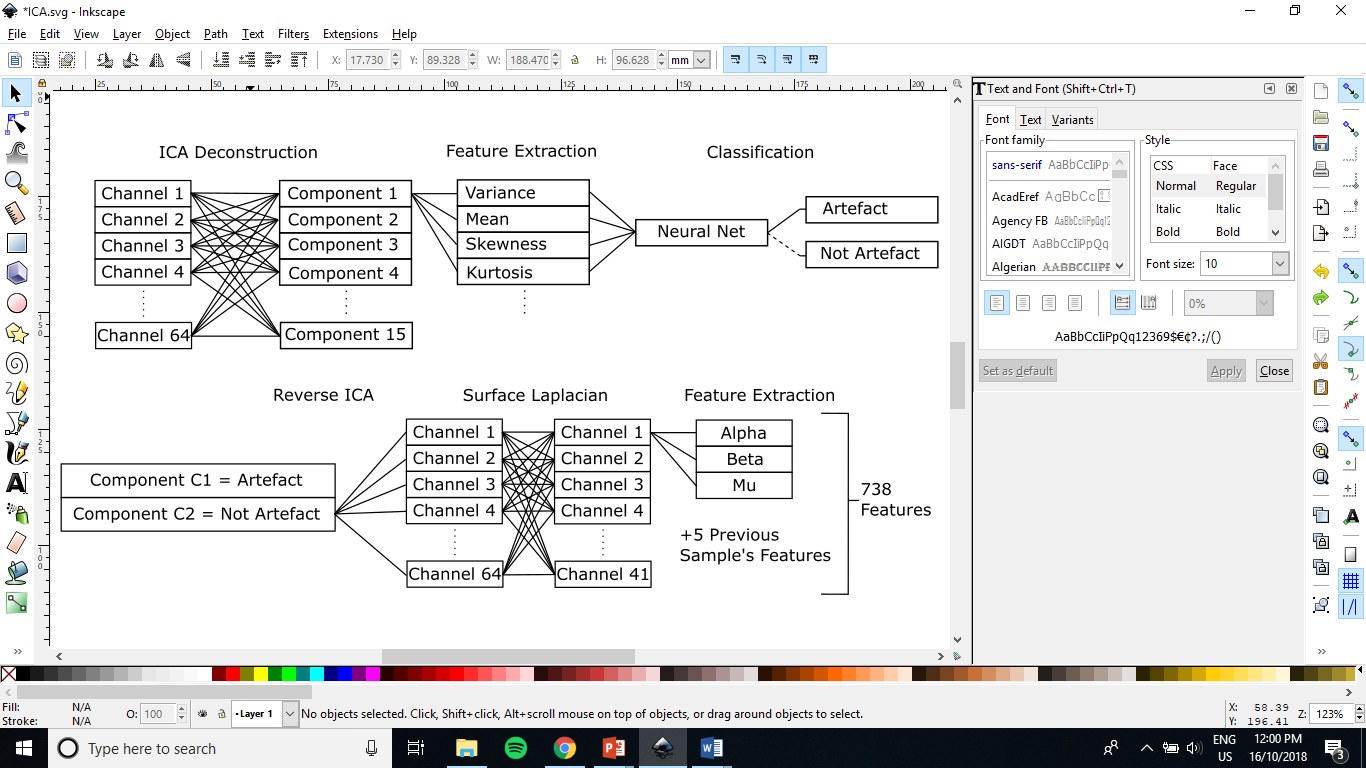
## Initial Filtering

Before anything else, we filter the data. We apply a bandpass filter between 0.5-80Hz to eliminate any baseline drift and high frequency noise. We also apply a 60Hz notch filter to reduce mains coupling (USA frequency). The frequency responses are shown in Figure X. Having performed basic filtering, the data is ready for artefact removal.

– filter frequency responses 😊

## Automated Artefact Removal

The AAR pathway is illustrated in full in Figure X. The following sections will discuss each operation chronologically and in detail.



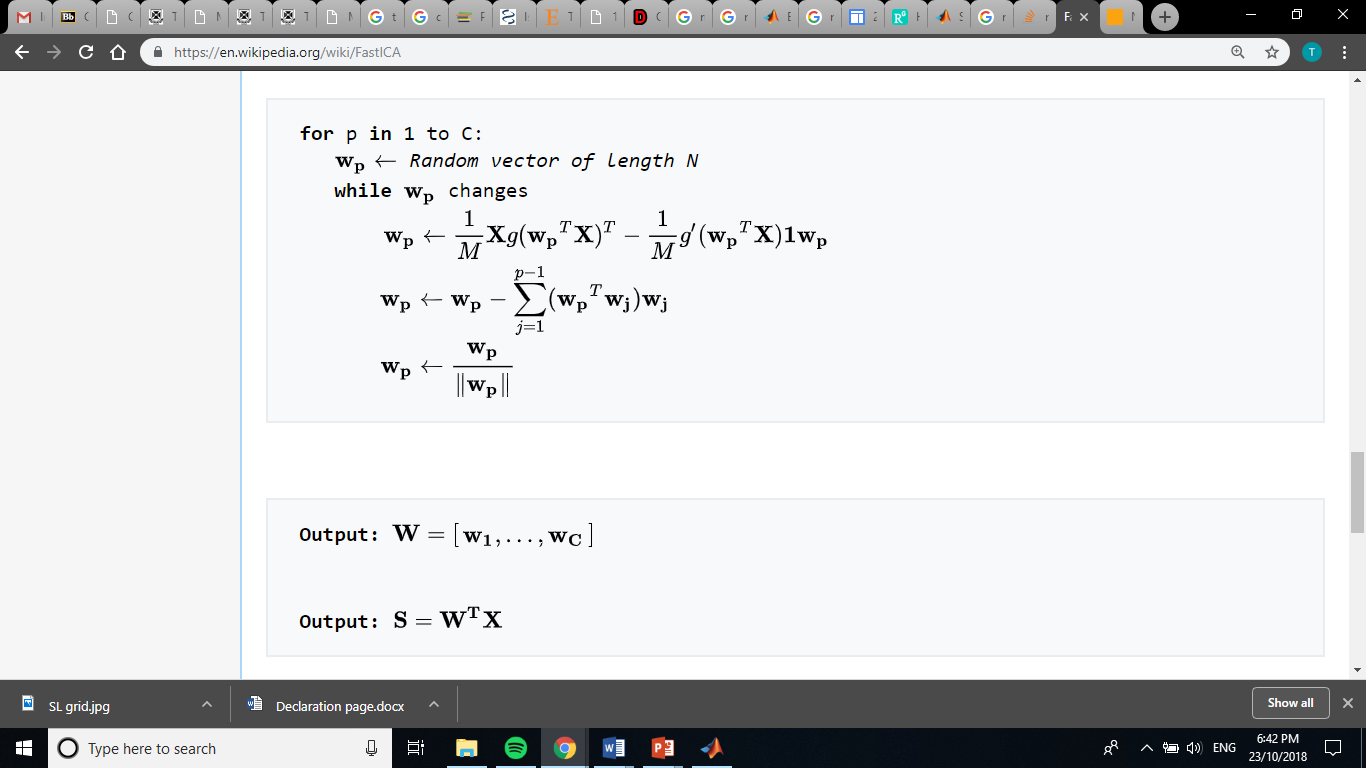
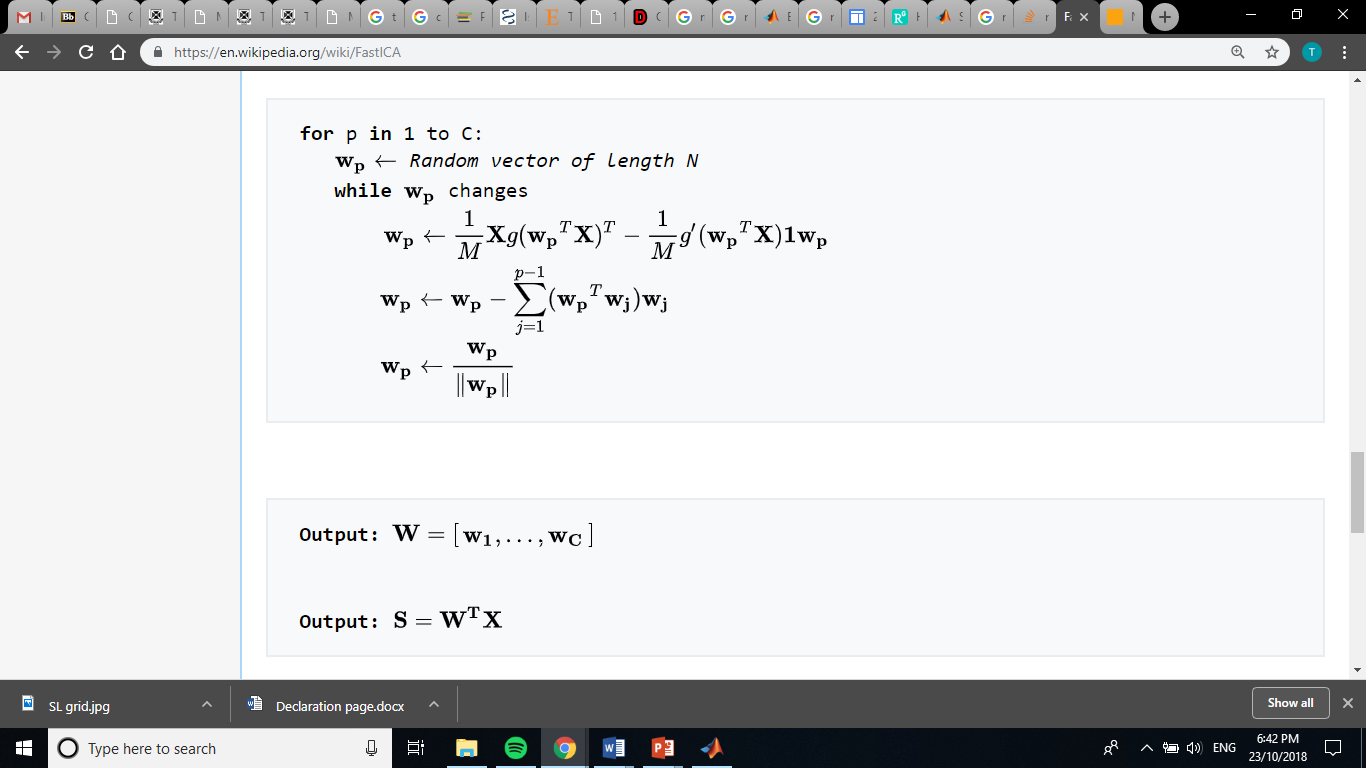
## Independent Component Analysis

ICA is applied to the data using the FastICA algorithm [21]. First the data must be centered and whitened. Centering the data implies setting the mean of each channel across all samples to zero:

Whitening the data involves transforming the centered data into an uncorrelated set with variance equal to one. This can be achieved through eigenvalue decomposition of the covariance matrix of the centered data, such that:

Where *E* is the matrix of eigen vectors and *D* is the diagonal matrix of eigenvalues.

With our nicely centered and whitened data, we can now apply the FastICA algorithm as shown in Figure X.



Where:

*C* is the number of components

*M* is the number of samples

*N* is the number of features

*W* is the component weight vector or unmixing matrix

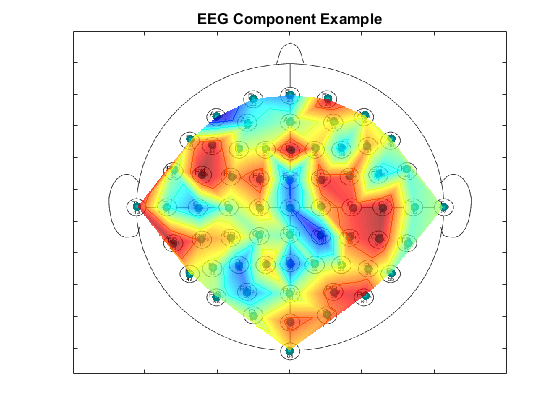
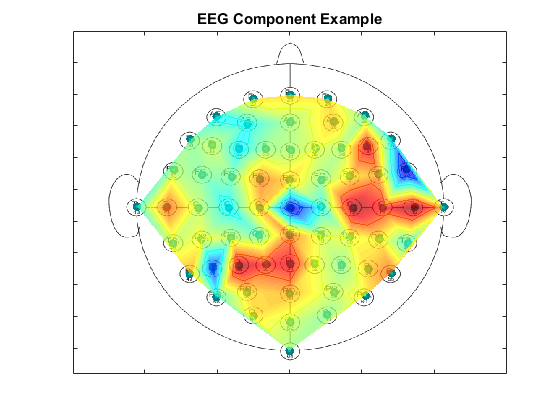
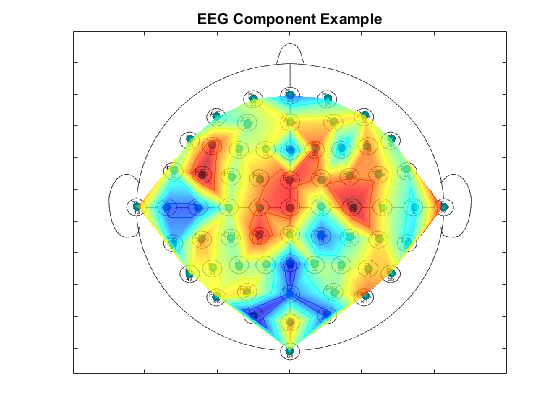
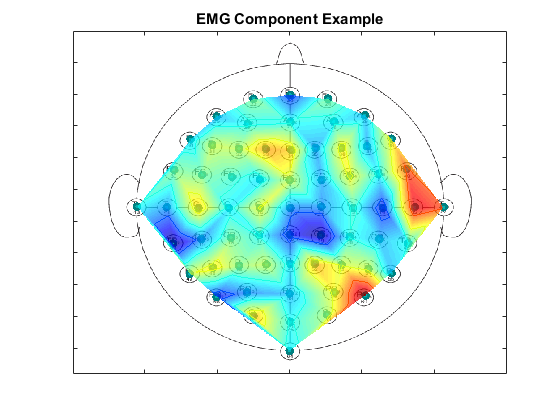
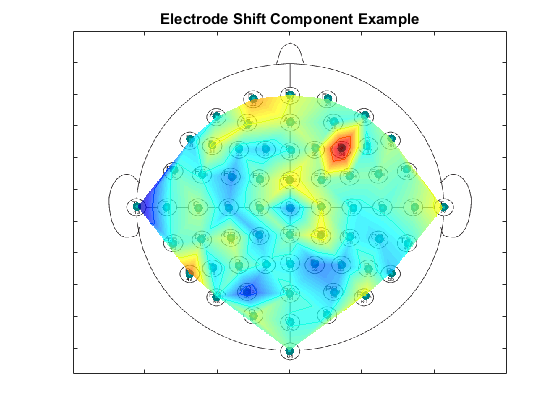
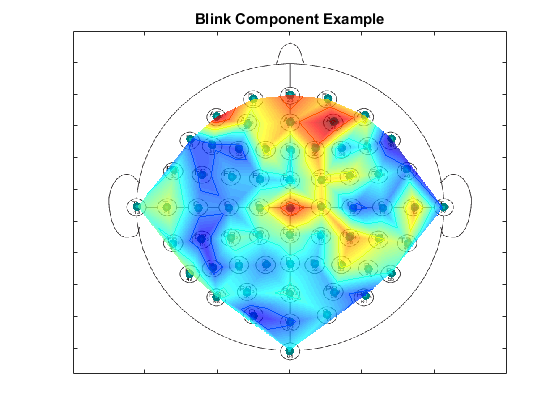
*S* is the component matrix

This algorithm is quite simple to implement in MATLAB. In the implementation used for this project, we consider the first 15 (C=15) components. The resulting *S* matrix gives us our component set. The next step is to extract features from these components to pass to a classifier. We can speculate what approach to take for feature extraction by examining what experts look for when manually labelling artefact components.

Two PhD candidates studying psychology and neuroscience at the University of Queensland, Brendan Keane and Nick Bland, have extensive experience analysing EEG data and identifying artefact components with ICA. Both were consulted to help identify signature attributes of an artefact component for the purposes of this project. Attributes indicating an artefact were summarised as:

1. Components which are spatially condensed to one, or few sparse channels (electrode shift)
2. Components with a frequency response showing peaks at high frequencies (>80Hz) or at USA mains frequency (60Hz)
3. Components with activity localised at peripheral electrodes (EMG) or frontal electrodes (EOG, blinking or eye movement)
4. Components spread over a strong majority if not all the channels (noise)

In the absence of all these observations, a component was labelled as *not* an artefact. Using these guidelines, 3000 components were manually labelled to be used as training data for a classifier to implement AAR. Examples of these components are displayed in Figure X.



## 3.2.2 Component Feature Extraction

Features were extracted from the components to provide the classifier with the tools to establish similar guidelines. The features are listed in Table X.

|  |  |
| --- | --- |
| Feature | Reasoning |
| Spatial mean of all channels | These features are tailored to inform the classifier of spatially observable artefacts such as those in described by guidelines I, III and IV. The classifier should learn that the amplitude of these features, and variations in comparative amplitudes of some these features can indicate that a component is or is not likely to be an artefact. |
| Spatial mean of peripheral channels |
| Spatial mean of central channels |
| Spatial mean of front channels |
| Spatial Variance |
| Spatial Kurtosis |
| Spatial Skew |
| Frequency response mean 0-80Hz | Features concerning the frequency response are aimed at arming the classifier with knowledge to identify components with high frequency or mains coupling noise, as described by guideline II. |
| Frequency response mean 59-61Hz |
| Frequency response mean >80Hz |

Feature-feature plots displaying the separation of the labelled data are shown in Figure X. The trends observable support the decision to extract such features for classification. The classifier implemented at the center of the AAR algorithm was an ANN.



## Artificial Neural Network

The ANN was initialized, trained and tested using code generated from the MATLAB neural network toolbox. The data was separated into train/validate/test datasets (70/15/15%). The network training function used was MATLABs inbuilt Bayesian Regularisation Backpropagation (trainbr). The stopping criteria consisted of the first to occur between: 10,000 epochs, gradient of < 1e-7 or 15 consecutive validation fails. The learning rate was set to 1e-4. The performance metric used is mean squared error. The layout of the network (ie. Number of hidden layers and perceptrons per layer) was tuned using a simple randomised search algorithm:

1. Randomise the number of perceptrons in a three-hidden-layer network between 0 and 8 per layer
2. Train and test the network
3. Keep the network if it produces the *best* *performance* thus far
4. Repeat from step 1.

It should be noted that while the performance metric used by the MATLAB toolbox for training the network was mean squared error, the metric used by the above algorithm in comparing two given networks was the mean between the sensitivity and specificity of the network. Sensitivity and specificity provide information regarding the capability of a classifier should the two classes have different prevalence within the data. In the case of artefact/non-artefact classification, 77% of the samples are labelled as non-artefacts. A classifier could easily achieve 77% accuracy in the trivial case where it classifies *every* sample as a non-artefact, thus classification accuracy alone is not an appropriate performance metric. Carefully considering the metric used has forced this simple algorithm to select networks which may sacrifice successful classification in the more prevalent class to improve classification in the other. This algorithm was left to run for several hours before settling for the current best layout, as shown in Figure X.

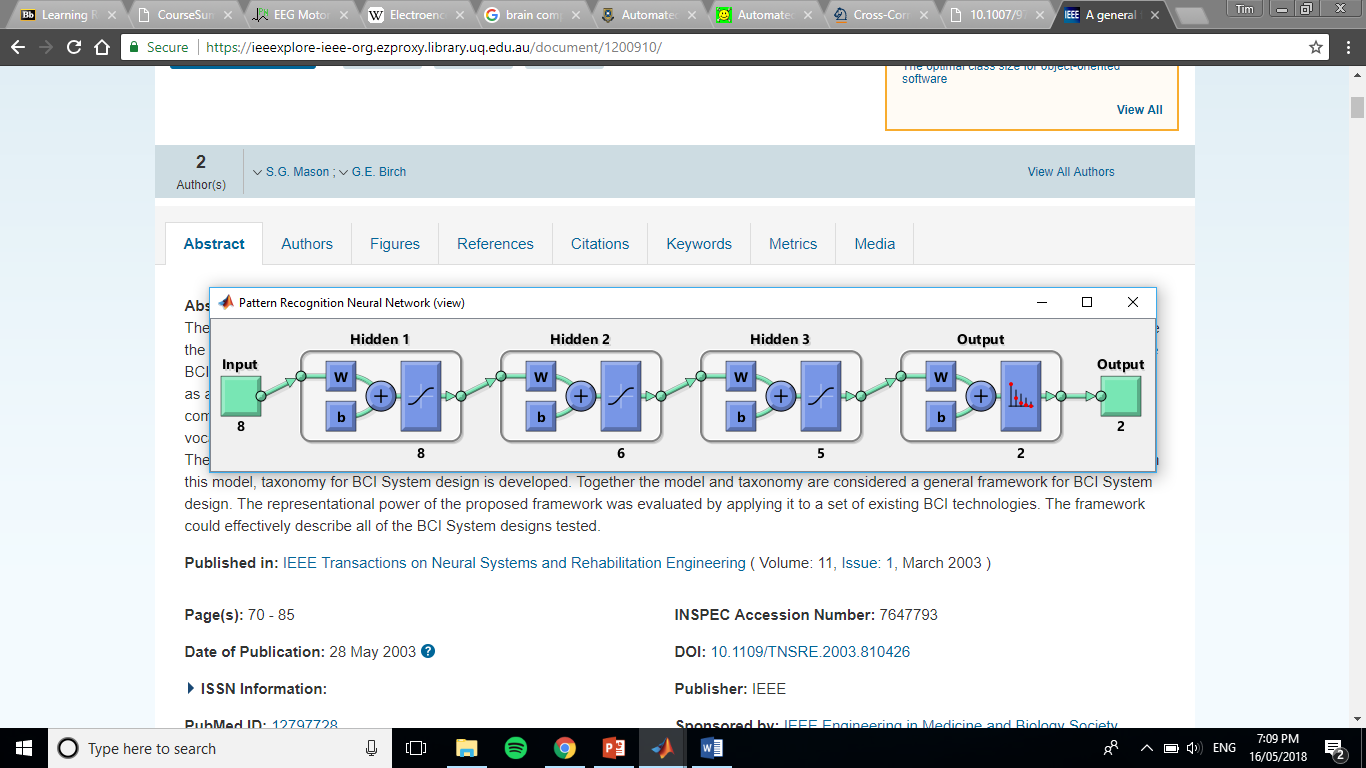


Figure X displays the confusion matrix for the network. The network produces reasonable performance (sens=95%, spec=80%). It is approximately 15% better at classifying non-artefact components (Class 1) than artefact components (Class 2). These results are considered reasonable as we would prefer to include data at the risk of it containing artefact, rather than exclude data which could potentially be useful to the BCI. An example of pre and post-AAR filtered data can be viewed in Appendix B.

## Channel Reconstruction

After classifying artefact components, the process of reconstructing the 64 EEG channels from the 15 detected components minus any artefacts is quite simple. We set the rows in the *S* matrix *component matrix* as per Figure X) to zero to create what we denote as *Sfiltered*, and recompute the mixed data *X* with the inverse of *WT*, as such:

The resulting *Xfiltered* matrix is our artefact removed 64-channel data, thus concluding the AAR algorithm.

## Surface Laplacian

With our filtered data we apply SL. Applying SL is a simple matrix multiplication, such that:

However, developing the *L* matrix requires applying the Finite Difference Method as shown in Appendix A and by Carvalhaes and de Barros [13] for each channel being included in the final feature set. We only consider the central 41 channels and ignore the peripheral 23, thus *L* becomes a 41x64 matrix. Each row is relating a channel to a scaled combination of only its close neighbours, hence *L* is mostly zeros. The MATLAB script .m file containing the *L* matrix initialisation is included with this thesis (formLS.m). Having performed the SL transformation, we are now ready to extract our features for classifying MI events.

## Feature Extraction

As discussed previously in the *Background* section (2.1.1, 2.1.2 and 2.1.3) we can utilize prior knowledge regarding MI neural activity to limit our feature search space in the time, space and frequency domains. We have already reduced the spatial dimension of the data by performing SL and only considering the central 41 channels. Regarding EEG frequency bands, we are only concerned with those typically relating to MM/MI neural events, ie. mu, alpha and beta. These frequency bands are attained using 10th order Butterworth bandpass filters at the thresholds listed in Table X. Including temporal patterns within the set allows for the capture of ERD/ERS events which have been previously shown to relate to MM/MI activity. These patterns generally occur over a period of seconds. Thus, we include 2.5 seconds worth of data in each feature set, updated and shifted along with every new sample at 2Hz. If our feature set is given by:

Then each is a single and unique combination of the frequency, time and space dimensions. In this case, (41x3x6). Table X illustrates the exact ordering of the feature string.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Frequency | Mu | | | | | | | Alpha | | Beta | |
| Channel | 1 | | | | 2 | … | 41 | 1 | … | | 41 |
| Time | t | t+0.5 | … | t+2.5 | t | t+2.5 | t | t+2.5 |
| Feature |  |  |  |  |  |  |  |

Before extracting these features however, we carefully consider how to segment the data. We have continuous recordings (160Hz) which we are subsampling in 2Hz blocks. We also have labelled data providing the exact starting time of the subject being given the cue for an MI event. This presents two options:

1. We specifically segment a temporal block about the exact time of the event (cherry picking) resulting in more consistent data for classification or
2. Sample the entire recording at 2Hz (as a BCI might in an online scenario) and choose epochs about the event, regardless of potentially leading or lagging.

While option I appears tempting, it does not reflect the capability of an online BCI. Segmenting the data in this way implies that the BCI can filter, perform AAR, extract features and classify new data at the same sampling frequency as the EEG hardware (160Hz in this case). This is not feasible with the proposed methods. Thus, we choose option II, which could result in the event leading/lagging the sample as much as 0.25 seconds, potentially causing inconsistencies between the same features for different samples. A BCI system which could perform well despite such a limitation would be a robust solution, satisfying the of the aims of this thesis. It is worth noting that at this point in the process, we leave out 20% of the data to use as a test set for classification later. We cannot use the entire dataset to first optimize our feature set, and then reuse the set for training a classifier as the system has already seen the labels, hence leading to overfitting.

## Feature Selection

Feature selection involves reducing the number of features to utilize relevance and minimize redundancy in the feature set. The problem can be modelled as binary optimization across *n* dimensions such that our feature set :

Where *n* is the total number of features. To search this unconventional space, we can employ the heuristic search method Genetic Algorithm.

## Genetic Algorithm Implementation

The general framework of GA is relatively constant for most applications: some form of randomised initialisation followed by selection, crossover and mutation operations. However, beyond this there are several degrees of freedom regarding hyperparameters and specific techniques used. Most variable criteria either present a trade off or are more applicable to certain types of problems than others. For instance, having a large candidate pool spans the search space more per generation, but takes longer to compute. K-point crossover is likely to preserve neighbouring genes through generations whereas uniform crossover will arbitrarily pass down from either parent at each gene. Too little mutation will slow the algorithm in terms of exploring potentially unseen features, but too much will undo the exploitative action of crossover and degrade the algorithm essentially into random search. These are all examples of design decisions which influence the performance and efficiency of a GA implementation. Through preliminary experimentation, we determined the specific GA method below to be appropriate for this feature selection problem.

1. Eight candidate sets are sampled at random from the feature space to create the *candidate* pool.
2. Evaluate *candidate* pool scores.
3. Automatically include the two highest scoring candidates (elite children) in the new *candidate* pool
4. Create a *mating* pool which includes two copies of the highest scoring candidate and one of each of the next two highest scoring candidates.
5. Select six sets of two different parents from the *mating* pool at random and perform uniform crossover to produce the remaining six candidates for the *candidate* pool.
6. Apply mutation with a probability of 0.01 to every candidate except for the best.
7. Repeat from step 2 until reaching a stopping criterion of 1000 generations.

Reasoning for each design choice is provided in Table X.

|  |  |
| --- | --- |
| Design Choice | Reasoning |
| Eight candidate generation | Eight appeared to be a healthy medium. Eight candidates span the space quite well allowing mutation to search a reasonable number of feature changes per generation whilst also including a good number of crossover combinations. One generation using a wrapper takes approximately four seconds (fewer using filter). |
| Two elite children | Including elite children solidifies high performing candidates into future generations lest they be improved upon. Future generations are encouraged to exploit and build on these feature sets through crossover. |
| The mating pool | Four candidates: two of the best, one of the second, one of the third. This architecture seemed to carry combinations to next generations from high performing candidates without resulting in homogeneity. |
| Two different crossover parents | Obviously different parenthood was enforced to discourage exact clones appearing in subsequent generations. Generations containing copies of candidates (especially high performing candidates) negates the benefits of the crossover operation. |
| Uniform crossover (as opposed to the more common k-point crossover) | There appears to be no reason to group neighbouring feature strings together during crossover. The inclusion or exclusion of each feature is seen as equally as important as any other feature on its own. For this reason, it seems more appropriate to pass on genes with equal probability from each parent at the gene to gene level. |
| Mutation probability of 1% | This parameter was decided through experimentation. Considering a string length of 738, 1% mutation appeared to provide enough variation for searching new solutions through the generations. |
| 1000 generation stopping criterion | This parameter is simply a time constraint. 1000 generations takes a little over an hour to compute. More generations could lead to a better solution with the cost of more computation time. In general, after several hundred generations, candidates with improved performance were less likely to be found. |

Initially, GA was implemented as a filter feature selector

## Filter Implementation

As discussed in the *Background* section (2.3), filter methods select feature sets using some fitness metric independent of a classifier. These methods are generally faster than wrapper methods, however may result in lower classification performance as it is not specifically selected for the classifier. In this project, we consider Correlation Feature Selection, which aims to select features with high correlation to the labels (*relevance*) and poor correlation with each other (*redundancy*) [16]. The score balances relevance and redundancy as such:

Where *k* is the number of features, is the mean correlation between the features and the class labels and is the mean inter-correlation between features. We aim to maximize this function as we require features which correlate strongly with the class labels, and poorly with each other.

We can exhaustively evaluate the correlation between features and the labels as well intercorrelation between features as this computation is not too taxing. Given that we have precomputed this information, calculating the score of any feature set is very fast. Before considering GA, we first consider the effectiveness of a simple algorithm to provide a benchmark: Greedy Forward Selection (GFS). The GFS algorithm is as follows:

1. Start with an empty feature set
2. Compute fitness of new *candidate* sets given the addition of a single feature.
3. Next set is the set from *candidates* with the best computed fitness
4. Repeat from 2 until all features are included in the set
5. Select the feature set which best balances fitness and number of included features

The score given by this algorithm over its progression is illustrated in Figure X. We simply choose the feature set with the highest score as it also yields a reasonably low number of features: 57 reduced down from the original 738.

The correlation score at this point is 0.2936. It is worth noting that this score is not bound between 0 and 1 as with correlation. In fact, its upper limit is dependent on the number of features included, as per the score equation. Given we decide to include 57 features, the Correlation Score space would appear as shown in Figure X, for varying mean relevance and redundancy correlations.



In the case of a perfect solution (,) we achieve a score of 7.5498 (). The best solution found using GFS is marked on this surface by the red asterisk and is far from the optimal solution. The algorithm was able to select a set with low mean redundancy, but also low mean relevance (, ). This poor score could indicate one of two things: that the features poorly represent the classes, given that the correlation score is a ‘good’ measure of this relationship, or that GFS is not an appropriate search algorithm for this problem. At this point, we hope for the latter, and that implementing GA to search for a better feature set can prove this.

Unfortunately, GA is not able to perform as well as GFS. Figure X displays the gradual increase in correlation score over the computed generations. Even having increased the stopping criteria to 10,000 (from 1000), GA was unable approach the best score found through GFS. A summary of the results observed using the two methods is shown in Table X.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Number of Features (*k*) | Redundancy (*rff*) | Relevance (*rcf*) | Correlation Score | Classification Error |
| GFS | 57 | 0.0364 | 0.0670 | 0.2936 | 0.4011 |
| GA | 249 | 0.0353 | 0.0407 | 0.2054 | 0.3671 |

While redundancy between the features is quite low for both methods, relevance is still far too low to assert any significant relationship between the features and the classes. Despite optimizing for correlation, the classification error yielded by each feature set when passed to a tuned SVM[[1]](#footnote-1) is quite poor. In addition, the set found with GFS resulted in a higher error when passed to a classifier than the set found using GA, despite a better correlation score. It seems a higher correlation score does not necessarily indicate better classification performance in this case. Perhaps a filter feature selector with a correlation metric is not an appropriate method for this particular problem. Taking this poor performance into consideration, we decide to change our approach and develop a wrapper feature selector with the hope of reducing classification error by tuning our feature set to a specific classifier.

## Wrapper Implementation

Unlike filter feature selection, a wrapper relies on the performance of some classifier as the metric for some set of features. The classifier being used for this feature selector is an SVM. Tuning the SVM involves appropriately selecting parameters gamma and *C* (as discussed in *Background* Section 2.3). The process for tuning for these parameters is suggested by MATLAB [23] and is outlined below:

* 1. Use MATLABs fitsvm function with default parameters to determine initial gamma.
  2. Adjust gamma and *C* parameters (KernelParameters.KernelScale and KernelParameters.BoxConstraint in MATLAB) through 11 values. Gamma consists of the default gamma found in step 1 scaled by 1e-5 to 1e5, increasing by a factor of 10. *C* is adjusted by starting with 1e-5 and increasing by a factor of 10 to 1e5. At each iteration we perform a 10-fold cross validation and retrieve the 10-fold loss.
  3. The best parameters of the 121 combinations is that which yields the lowest error.

The feature set used for this tuning was the set found previously using the correlation filter with GA. The parameters found to give the lowest error were gamma≈37.5 and *C*=1000. The kernel function used was a gaussian kernel. Having tuned the classifier, we are now ready to implement a wrapper feature selector.

To thoroughly investigate the potential of GA as a feature selector wrapper, we develop four methods:

1. **Greedy Backward Elimination (GBE)** – similar to GFS, GBE begins with a full set of features and greedily eliminates features which result in the lowest reduction in performance, as the name suggests. The progression gives us the freedom to choose at what number of features we believe the performance of the set to be satisfactory. However, as it is a greedy algorithm, we expect more sophisticated methods to perform better due to the complexity of the problem. GBE is used as a benchmark to compare our three remaining methods involving GA.
2. **Genetic Algorithm** – implemented in the exact same manner as described in *Methods* Section 3.1.1 with classification error being the feature set performance metric. It is worth noting that GA does not attempt to reduce the number of features in the set, which is one of the key aims of this thesis, hence the remaining methods attempt to enthuse feature reduction.
3. **Penalised Genetic Algorithm (PGA)** – we adjust the score metric to include a penalty term proportional to the number of features included in the feature subset, as per the equation below:

If our candidate set is given by:

Then PGA evaluates the error score of the set with the function:

Where *CE* is the classification error given by the train SVM and *R* is some scalar.

The value of *R* is representative of how much we penalise the set for including any given number of features. In fact, the equation implies that should we increase the number of features in the set by one, we would add *R* to the score. Thus, should we want to include some new feature in any candidate set, its inclusion should reduce *CE* by at least a value of *R* for it to be considered a worthy addition. Large values of *R* imply that we strongly oppose large feature sets and care little for classification error in comparison, whereas the opposite is true for small *R*. Selecting an appropriate value is key to the outcome of GA.

* 1. **Genetic Algorithm with Greedy Backward Elimination (GAwGBE)** – first GA is used to find a suitable feature set (as with method 2). The resulting feature set is then the starting point for GBE. This method aims to utilise the heuristic search of GA and then reduce the set using GBE, with the added benefit of allowing us to select how many features to eliminate before we cause too much error in classification.

The performance of these methods is presented in the *Results* section.

# Results

The following section presents the results of the four wrapper feature selection methods outlined in *Methods* Section 3.5.3. Data is separated into a training and test set prior to feature selection begins so as not to overfit the classifier. Classification error against the test data is provided in Section 4.4.

## Greedy Backward Elimination

Figure X illustrates the progression of GBE. As features are eliminated, we can observe the classification error reducing. This is likely due to many features being included in the set which do not relate to the classes, and hence the trained classifier poorly generalizes the classes in terms of these features. As such features are eliminated the classification error reduces to below 24% before increasing again as the feature set becomes too small to appropriately separate the classes. We choose the point marked by a red circle as the *best* solution provided by the algorithm as it has the lowest classification error and a significant number of eliminated features. Despite a reasonable solution, we hope to do better with the remaining, more sophisticated methods.

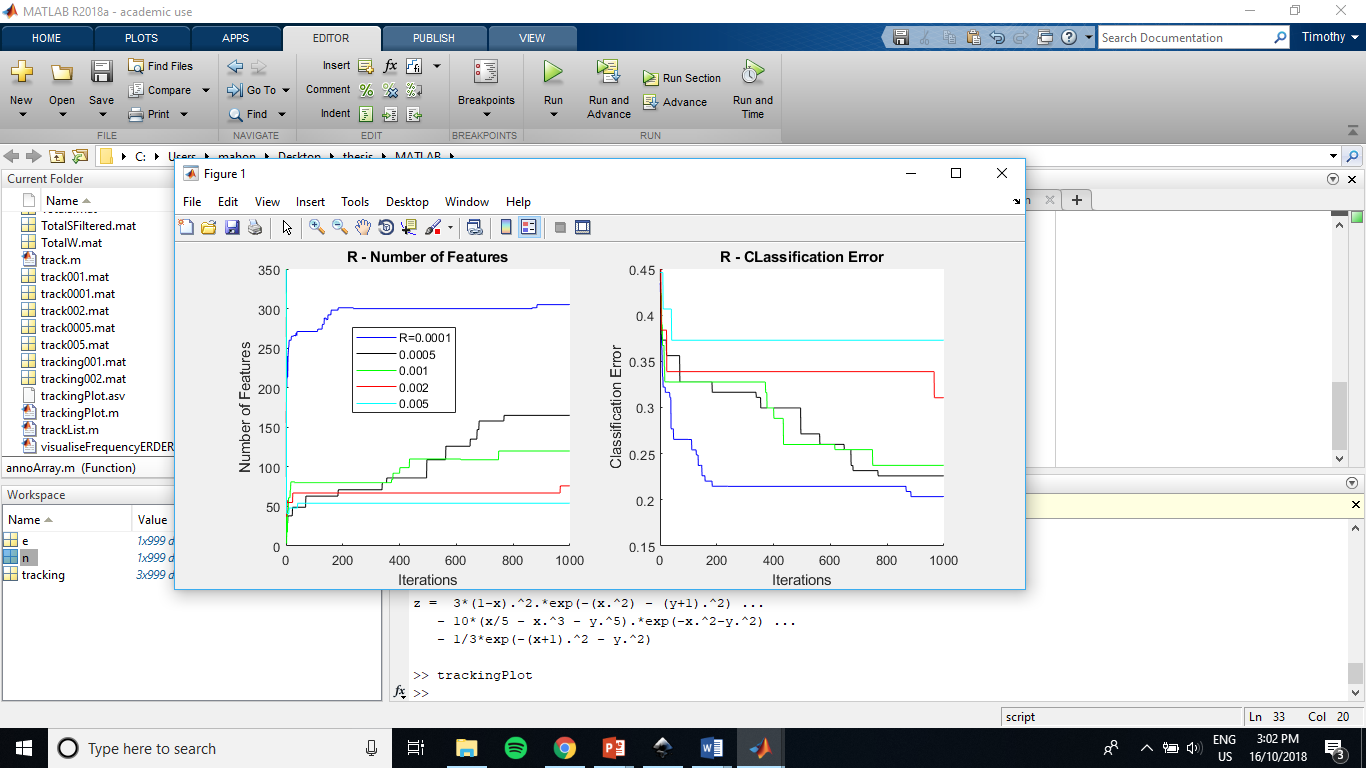


## Genetic Algorithm

Figure X displays the classification error and number of features given by the best candidate found using GA up to its stopping criteria of 1000 generations. We were able to achieve a lower classification error than the set provided by GBE, however the number of features in the set is over 300. We can also see that the reduction of classification error slows considerably past the first 100 generations, indicating that better solutions become increasingly sparse beyond <25% classification error. With the remaining two methods, our goals are to improve upon this solution by reducing the size of the set but maintaining reasonable classification error.

## Penalised Genetic Algorithm

Penalised Genetic Algorithm resulted in limited improvement upon standard GA. Figure X depicts the classification error and number of features for various values of *R*. It is clear that *R* behaves in the way described in *Methods* Section 3.5.3. As *R* increases, we can see the number of features in the candidate set drops over the generations, however this comes at a clear cost of classification error. Our best solution in terms of classification error is given by our smallest *R* (=0.001) with an error≈0.2 but over 300 features. Whereas when *R*=0.005 we have a set with almost 50 features but an error>0.35. Of those trialed, there appears to be no overwhelming *R* values which produces a solution significantly better in terms of both of these parameters, however for the sake of comparison we consider the case where *R*=0.001 to be a reasonable medium.



## Genetic Algorithm with Greedy Backward Elimination

Genetic Algorithm followed by Greedy Backward Elimination produced the best solution of the four methods in terms of both classification error and number of features. Figure X illustrates the progression of the two algorithms. GA finds a set with a reasonable error but high number of features, as shown previously in the results of method 2 (Section 4.2). The addition of GBE then produces a linear decrease in the number of features while maintaining, and for the majority of the progression *improving*, the error rate of the SVM. Indeed, in this case GBE was able to eliminate over 200 features from the set found with GA before error starts to increase. The combination of these two algorithms appears to have performed better than both alone, and the proposed altered method PGA.



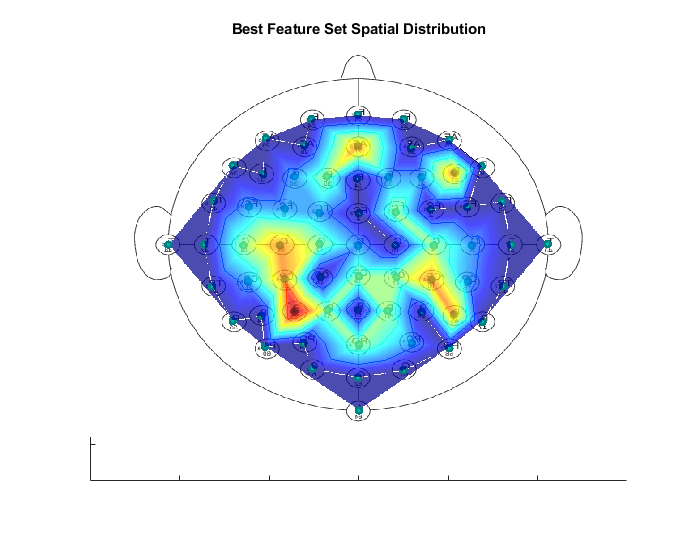
## Summary

Table X displays a summary of the performance of the methods for ease of comparison. We include the PGA score for the best candidate found using each method given *R*=0.001.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Classification Error (Test Data) | Number of Features | Score (R=0.001) |
| GBE | 0.2350 | 80 | 0.3150 |
| GA | 0.2091 | 328 | 0.5371 |
| Penalised GA | 0.2410 | 131 | 0.3720 |
| GAwGBE | 0.1864 | 52 | 0.2384 |

## The Feature Set

While no predictions or preferences are in place regarding what specific features are included within the reduced set, it may be interesting to examine their distribution across the time, space and frequency domains and briefly comment on any apparent trends. This distribution is depicted in Figure X.



Regarding the frequency distribution, all the included bands (mu, alpha and beta) are often related to MI events and hence it is no surprise that all are included to some degree. While beta band being significantly more present in the set than the other two bands is interesting, any conclusions which could be drawn from this are not the focus of this thesis and are hence outside of its scope. The wide spread of temporal features presents a limitation for the use of this set as a BCI system: temporal information required for classification exists as far as 2.5 seconds after the MI event. This results in a lag in the classification. Reducing this lag could be an area of further research, as discussed later in Section X. The spatial distribution as seen in Figure X.X depicts a certain likeness to what one might expect given the mirrored channels approximately clustered about the *hand* section near the motor cortex. Recalling the cortical homunculus discussed in *Background* section 2.1.1, we might expect to see channels localized around the motor cortex (approximately central horizontal band) and slightly offset from the vertical center, which is marginally similar to what we see in the figure. Stipulating further than this is, again, not the focus of this thesis. It is important to note that the number of channels has been reduced from the original 64 down to 28, satisfying one of the aims of this thesis by reducing hardware requirements for such a BCI system.

# Discussion

This section aims to critically analyse the results presented, particularly focusing of the effectiveness of the wrapper methods as they are considered the prominent center of this thesis. Beyond discussion of the results, we include a reflection on the effectiveness of risk management methods employed as per the original project proposal, as well as the project plan and any variations to it over the course of this thesis.

## Filter Feature Selector Performance Review

The filter feature selection method performed poorly. We observed not only that both GFS and GA could not find a feature set with significant relevance to the classes, but also that when it came to classification error, the set found using GA produced lower error despite having a lower correlation score. These observations are not in line with what we might expect but can be explained quite simply: correlation is not a suitable metric to evaluate this particular problem. Correlation appears to poorly represent the fitness of a given feature because the measure being used in this case (Pearson Correlation Coefficient) is the strength of a *linear* relationship between two variables. While it is difficult to illustrate the relationship between each feature and their labels given the number of dimensions, the inherent complexity of the MI event classification problem implies that this relationship is indeed non-linear. Hence employing a linear metric to asses our feature set’s fitness seems to be an inappropriate approach, a conclusion which is supported through the experimental results. The one advantage the filter method had over wrappers was the significantly reduced computation time, however this benefit is null considering its poor performance. A wrapper method including a classifier capable of separating non-linear classes, such as an SVM, provides a more suitable solution.

## 5.2 Wrapper Feature Selector Performance Review

First let’s consider the basic methods: GBE and standard GA. GBE performed as expected, providing the freedom to choose when to cease the elimination and accept the classification error of the reduced set. The method provided us with far superior classification performance than the filter implementations (error=0.23 vs 0.40/0.37), supporting the choice to wrapper feature selection. The error it produced was low, but high enough still that we hoped for further improvement through more sophisticated heuristic search methods. Lower error was indeed found with standard GA (0.21 vs. 0.23). This confirmed two things: the solution found by GBE was not globally optimal with respect to classification error, and that our particular GA implementation is capable of finding better solutions. The former point may have been stipulated prior to this discovery, but its confirmation means that the introduction of sophisticated search algorithms such as GA to solve this feature selection problem is justified. The latter point, that the GA algorithm design as described in *Methods* Section 3.5.1 is indeed valid, attests to the success of this thesis in applying such an optimization technique in this context. This is a key milestone in the project, however one aim remains to be included in the optimisation method, and that is encoding the reduction of features.

This encoding was achieved in two distinct ways by the remaining two wrapper feature selection methods. Recall PGA introduced a penalty term into the score metric, and GAwBGE selected a starting set for BGE using GA. Of these two methods, GAwBGE was able to achieve lower classification error *and* a lower number of features. We can stipulate as to why this has occurred. Regarding PGA, consider the case where our generation includes a candidate with a high scoring candidate with a reasonable number of features. This candidate is chosen as a parent for crossover and undergoes mutation, producing a child candidate. One of the following four scenarios may occur:

1. The child candidate includes more features and results in a higher error score
2. The child candidate includes more features and results in a lower error score
3. The child candidate includes fewer features and results in a higher error score
4. The child candidate includes fewer features and results in a lower error score

and recall the equation for PGA score,

We know that the score is proportionally related to the number of features in the set. Thus, with the inclusion of more features, it is more likely that we will result in a higher (worse) score. The reverse is also true. With this, we could say that scenarios 1 and 2 are more likely than 3 and 4 as they would require the child candidate to improve (or worsen) their score by a value more than the number of features gained (or lost) multiplied by *R*. Next consider that during the operations of crossover and mutation, there is no preference concerning the passing on or manipulation of a 1 bit over a 0. Yada yda

Despite all this the classification error wasn’t anything special

Compare with previous study

Why did this happen?

# Risk analysis

Discuss how successful risk mitigation was.

This risk analysis will briefly outline the three risks involved with this project, the level of risk and actions that will be taken to minimize this risk. The risk level will be assessed against the following risk matrix:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Severity | | | | |
| Likelihood |  | Negligible | Marginal | Critical | Catastrophic |
| Rare | Low | Low | Moderate | High |
| Unlikely | Low | Low | Moderate | Extreme |
| Possible | Low | Moderate | High | Extreme |
| Likely | Moderate | High | High | Extreme |
| Certain | High | High | Extreme | Extreme |

## Breach of Copyright

Copyright laws could be breached by including datasets and results from other studies in this project.

**Likelihood:** Rare

**Severity:** Marginal

**Risk level:** Low

**Mitigations:** All ideas and data used within this report which are not owned by the author will be referenced correctly in accordance with the University of Queensland referencing and plagiarism policy as well as per request by the original owner/s. In addition, datasets will only be collected from reputable sources.

## Data Loss

Unforeseen circumstances such as damage to or loss of the authors computer, data or other records could lead to loss of project progress. As the project progresses, the severity of this risk increases to a critical level.

**Likelihood:** Unlikely

**Severity:** Critical

**Risk level:** Moderate

**Mitigations:** All data and electronic records will be committed regularly to a git repository dedicated to this thesis project. This includes but is not limited to: any MATLAB script, function files or data sets, electronic notes and assessment items. Care will be taken to minimise risk of malware being downloaded. Computers being used to work on the project will not be left unattended while thesis material is able to be accessed.

## Underestimation of Time Requirement

This project includes many steps which require careful implementation and validation before advancing to the next stage, as discussed in Section 5. Incorrect implementation or underestimation of the time required to complete anyone of these stages could lead to limitations in the project due to time constraints.

**Likelihood:** Possible

**Severity:** Critical

**Risk level:** High

**Mitigations:** A well defined project plan has been outlined in the following section which includes milestones designed to motivate completion of the thesis within a manageable timeframe. This plan has been designed to expect delays and provide adequate time to overcome obstacles. In addition, regular correspondence with the thesis supervisor Prof. Gallegher or one of his tutors/associates has been agreed upon to verify progress and provide guidance if required.

# Project Plan

Compare original plan to actual

Discuss changes in the plan ie. Filter to wrapper, classifiers, PSO

Further research

# Conclusion

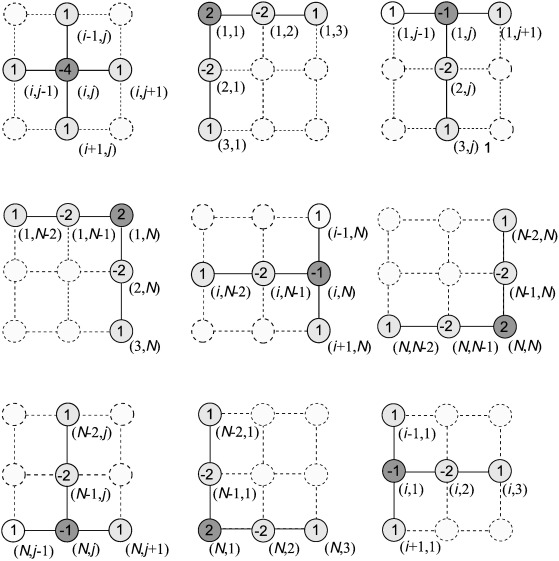
This proposal has clearly outlined the motivations, scope and aims of this project. Considerate background research was conducted to establish a solid understanding of the concepts involved across the disciplines of biology, engineering and computer science. Investigations into previous studies indicated that this particular focus on a filter type feature selector using PSO and GA for EEG MI/MM classification has not been conducted before to the authors knowledge. However, outcomes from similar studies provide confidence in this projects achievability as well as supplying a benchmark to compare results with. Potential risks to the project have been considered and steps put in place to mitigate them, and a detailed project plan has been implemented to ensure satisfactory progress. To conclude, this project is designed to explore useful techniques within the field of BCI development with the hope that further technological advances could lead to a viable alternative device control mechanism.

# Bibliography

|  |  |
| --- | --- |
| [1] | J. Constine, "Facebook is building brain-computer interfaces for typing and skin-hearing," Tech Crunch, 2017. [Online]. Available: https://techcrunch.com/2017/04/19/facebook-brain-interface/. [Accessed October 2018]. |
| [2] | "Brain-Computer Interfaces," JHU Neuroengineering & Biomedical Instrumentation Lab, 2008. [Online]. Available: http://neuroengineering.bme.jhu.edu/ecog.htm. [Accessed 22 March 2018]. |
| [3] | "Neuralink," Neuralink Corp, 2018. [Online]. Available: https://www.neuralink.com/. [Accessed 22 March 2018]. |
| [4] | A. L. G. L. H. J. I. P. M. R. M. J. M. G. P. C.-K. S. H. Goldberger AL, "EEG Motor Movement / Imagery Dataset, PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. Circulation 101(23):e215-e220," 13 June 2000. [Online]. Available: https://www.physionet.org/pn4/eegmmidb/. [Accessed March 2018]. |
| [5] | G. M. D. H. T. B. N. W. J. Schalk, "BCI2000: a general-purpose brain-computer interface (BCI) system.," *IEEE Trans Biomed Eng,* vol. 51, no. 6, pp. 1034-1043, 2004. |
| [6] | M. Schroder, M. Bogdan, T. Hinterberger and N. Birbaumer, "Automated EEG feature selection for brain computer interfaces," in *Conference Proceedings. 1st International IEEE EMBS Conference on Neural Engineering 2003*, Capri Island, Italy, 2003. |
| [7] | S. Suily, Y. Li and Y. Zhang, EEG Signal Analysis and Classification: Techniques and Applications, Gewerbestrasse: Springer International Publishing, 2016. |
| [8] | V. Towle, J. Bolaños, D. Suarez, K. Tan, R. Grzeszczuk, D. Levin, R. Cakmur, S. Frank and J.-P. Spire, "The spatial location of EEG electrodes: locating the best-fitting sphere relative to cortical anatomy," *Electroencephalography and Clinical Neurophysiology,* vol. 86, no. 1, pp. 1-6, 1993. |
| [9] | Y. Kim, J. Ryu, K. K. Kim, C. C. Took, D. P. Mandic and C. Park, "Motor Imagery Classification Using Mu and Beta Rhythms of EEG with Strong Uncorrelating Transform Based Complex Common Spatial Patterns," *Computational Intelligence and Neuroscience,* vol. Online, 2016. |
| [10] | D. McFarland, L. Minor, T. Vaughan and J. Wolpaw, "Mu and Beta Rhythm Topographies During Motor Imagery and Actual Movements," *Brain Topography,* vol. 12, no. 3, pp. 177-186, 2000. |
| [11] | A. R. Satti, D. Coyle and G. Prasad, "Spatio-spectral & temporal parameter searching using class correlation analysis and particle swarm optimization for a brain computer interface," in *IEEE International Conference on Systems, Man and Cybernetics*, 2009. |
| [12] | S. Mason and G. Birch, "A general framework for brain-computer interface design," *IEEE Transactions on Neural Systems and Rehabilitation Engineering,* vol. 11, no. 1, pp. 70-85, 2003. |
| [13] | C. Carvalhaes and J. A. de Barros, "The surface Laplacian technique in EEG: Theory and methods," *International Journal of Psychophysiology,* vol. 97, no. 3, pp. 174-188, 2015. |
| [14] | Y. Saeys, I. Inza and P. Larrañaga, "A review of feature selection techniques in bioinformatics," *Bioinformatics,* vol. 23, no. 19, pp. 2507-2517, 2007. |
| [15] | I. Guyon and A. Elisseeff, " An Introduction to Variable and Feature Selection," *Journal of Machine Learning Research,* vol. 3, pp. 1157-1182, 2003. |
| [16] | K. Deb, "An introduction to genetic algorithms," *Sadhana,* vol. 24, no. 4-5, pp. 293-315, 1999. |
| [17] | A. Subas and M. Gursoy, "EEG signal classification using PCA, ICA, LDA and support vector machines," *Expert Systems with Applications,* vol. 37, no. 12, pp. 8659-8666, 2010. |
| [18] | M. Bentlemsan, E. Zemouri, D. Bouchaffra, B. Yahya-Zoubir and K. Ferroudji, "Random Forest and Filter Bank Common Spatial Patterns for EEG-Based Motor Imagery Classification," in *Intelligent Systems, Modelling and Simulation*, Langkawi, Malaysia, 2014. |
| [19] | A. Subasi and E. Ercelebi, "Classification of EEG signals using neural network and logistic regression," *Computer Methods and Programs in Biomedicine,* vol. 78, no. 2, pp. 87-99, 2005. |
| [20] | M. Alomari, A. Samaha and K. AlKamha, "Automated Classification of L/R Hand Movement," *International Journal of Advanced Computer Science and Applications,* vol. 4, no. 6, pp. 207-212, 2013. |
| [21] | A. Hyvarynen and E. Oja, "Independent Component Analysis: Algorithms and Applications," *Neural Networks,* Vols. 4-5, no. 13, pp. 411-430, 2000. |
| [22] | R. Poli, J. Kennedy and T. Blackwell, "Particle Swarm Optimization," *Swarm Intelligence,* vol. 1, no. 1, pp. 33-57, 2007. |
| [23] | I. Koprinska, "Feature Selection for Brain-Computer Interfaces," in *Proceedings of the Workshop on Advances and Issues in Biomedical Data Mining, Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2009)*, Bangkok, 2009. |

Appendix

A: SL



Pre and post AAR filtered data example.

Brief description of each code function / .m file

1. Tuning of SVM classifier is described in detail in *Methods* section 3.5.3. [↑](#footnote-ref-1)