Identifying Users and Activities from Brain Wave Signals Recorded from a Wearable Headband

by

Glavin B. Wiechert

A Thesis Submitted to
Saint Mary's University, Halifax, Nova Scotia
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science (Honours)

April 2016, Halifax, Nova Scotia

Approved:	Dr. Pawan Lingras Supervisor
Approved:	Dr. Porter Scobey Reader
Approved:	Dr. Sageev Oore Reader

Dated: April 27, 2016

Abstract

Identifying Users and Activities from Brain Wave Signals Recorded from a Wearable Headband

by Glavin B. Wiechert

This paper studies the supervised classification of electroencephalogram (EEG) brain signals to identify persons and their activities. The brain signals are obtained from a commercially available and modestly priced wearable headband. Such wearable devices generate a large amount of data and due to their attractive pricing structure are becoming increasingly commonplace. As a result, the data generated from such wearables will increase exponentially, leading to many interesting data mining opportunities.

This paper proposes a representation that reduces variable length signals to more manageable and uniformly fixed length distributions, and then explores the effectiveness of a variety of data mining techniques on the biometric signals. The proposed approach is demonstrated through data collected from a wearable headband that recorded EEG brain signals. The brain signals are recorded for a number of participants performing various tasks. The experiments use a number of classification and clustering techniques, including decision trees, SVM, neural networks, random forests, K-means clustering, and semi-supervised crisp and rough K-medoid clustering.

The results show that it is possible to identify both the persons and the activities with a reasonable degree of precision. Furthermore, for identifying persons the evolutionary semi-supervised crisp and rough K-medoid clustering is shown to favourably compare with the conventional unsupervised algorithms such as K-means.

April, 2016

Acknowledgements

I would like to thank my supervisor, Dr. Pawan Lingras, for always challenging me, facilitating my academic interests, and providing the encouragement I need when I needed it the most. I would also like to thank Matt Triff for his invaluable assistance throughout the development of this research. Also, thank you to Zhixing Liu, Zhicheng Yin, Shuai Zhao, and Ziyun Zhong for their assistance with the data collection.

In addition, a thank you to both Dr. Porter Scobey and Dr. Sageev Oore, for sharing your knowledge and wisdom with me in the classroom and extending your support to be examiners of my thesis defence. I would also like to thank the Natural Sciences and Engineering Research Council of Canada and the Faculty of Graduate Studies and Research, Saint Mary's University for funding.

Finally, I would like to thank my family and friends – especially my partner, Celina Campbell – for their moral support and unwavering confidence in me.

Without each of you I would not be where I am today.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Wearable devices include wristbands, armbands, watches, headbands that record signals from different parts of the body. Typically, the devices record all the signals from the muscles they are in contact with or the brain, leading to a large amount of data. Both the volume and velocity of the data makes it a big data problem. It is not easy to analyze a long sequence of numbers recorded on different channels for meaningful data mining activities.

This paper presents data collection from a wearable headband called Muse. Among other variables such as acceleration and facial movements, the Muse headband records four signals from four different locations around a person's head. The data streams from the headband can be very long. It is necessary to create a more compact representation of the data streams from various channels without losing the essence of the data pattern.

This paper describes an approach to create a summarized frequency distribution of the signals. The approach is tested for a data collection consisting of five individuals performing a number of tasks; listening to music, reading, relaxing, watching a video, playing computer games, and listening to music. These activities are repeated a number of times to test the validity of the knowledge representation and subsequent data mining. The dataset with the proposed knowledge representation is used with a number of supervised classification [8] techniques including decision trees [1, 2, 34], support vector machines (SVM) [3, 4, 36, 37, 38], neural networks [5, 9, 40], and random forests [2, 10, 34, 35]. The variety of classification techniques demonstrated the ability of the summarized frequency distribution to represent the time series of signals independent of the classification techniques used. The classification process was successful in predicting the persons as well as the activities.

Clustering, a supervised learning technique, is one of the first data mining techniques applied to any new dataset to explore the patterns that may exist. A good clustering algorithm should minimize the within-cluster scatter and maximize the distance between clusters. Since there are n^k different possible clustering schemes for clustering n objects into K-clusters, a straightforward optimization is an NP hard problem. K-means is one of the popular clustering algorithms that can find a reasonably optimal clustering scheme. K-medoid clustering is a variation of the K-means algorithm. The K-means approach determines the centroid of a cluster, whereas K-medoid clustering finds an object that is most similar to all the other objects in the cluster. The search space for K-medoid is discrete and limited by the total number of objects in a dataset. This can be an advantage in evolutionary searching. The standard K-means and K-medoids algorithms focus on the distance-based metrics, such as within-cluster scatter and separation between clusters. It is not possible to include additional optimization criteria in these algorithms. Combining K-medoids with an evolutionary algorithm makes it possible to perform multi-objective clustering depending on the need of an application. Peters [11] first proposed the use of evolutionary computing in the context of rough set theory, Lingras [12, 13] subsequently explored both K-means and K-medoids based on rough set theory.

The clustering approaches group the data based on the distances between objects. They do not and cannot accommodate any other known categorization of these objects. In some cases, it is possible that we may have additional information regarding the categorization of some of the objects in a dataset. This paper extends the evolutionary rough K-medoid algorithm to optimize the precision of the known categorization in the dataset. The need and usefulness of the proposed approach is demonstrated for a dataset consisting of biometric signals from a wearable device.

The dataset is categorized using the K-means algorithm. The results are compared with the proposed evolutionary semi-supervised crisp and rough K-medoid algorithm. The results show that a repeated application of the proposed semi-supervised technique may be able to create a clustering scheme that can improve the precision of known category information.

Chapter 2

Literature Review

2.1 Wearables

Wearable technologies are evolving quickly and companies are discovering innovative ways to utilize the enormous amount of data to which they now have access. Experts estimate the Internet of Things (IoT) will consist of almost 50 billion objects by 2020 [14]. The IoT is a network of physical objects, or things, that are embedded with sensors and software that can collect, transmit, and receive data over a network. A common example of a wearable technology is a smart watch, such as the Pebble or Apple Watch. Such devices provide the portability and convenience of a watch and include an embedded computer that can connect to your smart phone to perform complex tasks, such as managing SMS messages, providing turn-by-turn navigation, and even perform commands activated by voice.

There are many different categories of wearables. Each category collects its own unique type of data that can be used for meaningful data mining activities. This paper will specifically investigate the collection of electroencephalogram (EEG) brain signals from a Muse headband.

2.1.1 Wristbands and Watches

Smart watches such as the Pebble or the Apple Watch provide an interface to notifications, such as messages, calendar events, or breaking news, that are typically viewed on a smartphone. Additionally, as a wearable they have their own sensors such as GPS, accelerometers, magnetometers, and heart rate sensors [15, 16]. These sensors provide additional capabilities for users, who can now track activity levels or health information.

Wristbands, such as Fitbit devices or the Nymi band provide more specialized functionality and typically do not rely on an ongoing connection to a smartphone. The Nymi band for example measures an individual's electrocardiogram (ECG) to create a unique biometric identifier that can be used in place of a traditional password [17]. Fitness trackers such as the Jawbone UP or Fitbit devices are wearable wristbands that can measure health and activity information. They can record information such as how long and how far the individual has performed a specific exercise, the duration and quality of their sleep, and heart rate levels. Fitness tracker wearables such as the Fitbit devices aim to motivate users and provide feedback through the measurements they gather [18, 19]. Fitness trackers often tie in to smartphone or online applications to better analyze the data they collect, and to provide feedback to users.

2.1.2 Armbands

Armband wearables, such as the Myo, use sensors to monitor the user's gestures and movements [20]. The Myo, for example uses electromyography (EMG) sensors to detect the electrical activity changes as the user activates different muscles and combines the EMG input with gyroscope, accelerometer, and magnetometer sensors [21].

2.1.3 Headbands, Headsets, and Smartglasses

Head-mounted wearables in general either augment the wearer's capabilities by displaying additional information through a heads-up display, as in the case of Google Glass or the Microsoft HoloLens, or use sensors to measure the wearer's brain activity, as in the case of the Emotiv and Muse headbands.

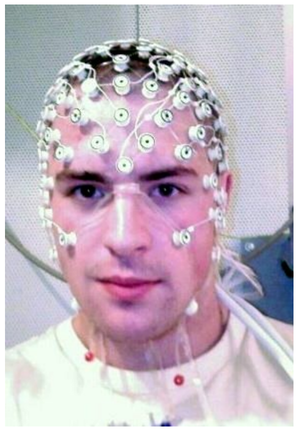
Figure 2.1: EEG Cap

The Google Glass headset provides a display and sensors to show the information found on a smart watch or smartphone. Additionally, a camera in the device can be used not only for taking pictures, but also for computer vision tasks such as object recognition or, in combination with the display, augmented reality where computer generated graphics are displayed as an overlay to highlight objects in the real world [22]. Other headsets, such as the Microsoft HoloLens are expected to provide a similar augmented reality experience [23], serving as a way to both improve creation and consumption of multimedia. The HoloLens also promises to improve interactions with real world objects, such as providing step-by-step instructions on repairing a light switch [24].

Headbands such as Muse or Emotiv provide brain-computer interfaces (BCI). The

Muse headband uses four EEG sensors, two on the forehead and two behind the ears, and three additional reference sensors [25]. Muse can also record blinking and jaw clenching. These sensors are used to detect and measure electrical activity in the brain. Currently the Muse headband is targeted as a mindfulness training device, helping users calm their minds. However, developers have used the device as a BCI to control robots and perform research. Similarly, the Emotiv makes use of even more EEG channels, accelerometer sensors, magnetometers, and gyroscope sensors to detect a wide variety of facial expressions, emotional states, and mental commands, such as push, pull, left, or right. [26, 27].

2.1.4 E-textiles – Smart clothing, smart textiles, smart fabrics

E-textiles comprise various clothing and fabrics with integrated electronics that allow them to communicate, measure and transform [28]. Smart clothing, such as the smart shirts developed by OMsignal and Hexoskin, provide in-depth biometric measurements including heart rate, breathing rate, step counts, calorie counts and more [29, 30]. The smart clothing is used by athletes to improve training and athletic performance, and by researchers to research sleep patterns, stress levels, respiratory ability, and air pollution [31].

Smart fabrics such as Smart Skin, while currently not a wearable for humans, are used for products on packaging lines to measure various factors such as pressure and orientation to provide higher levels of quality assurance [32].

Fabrics that enable electronics to be embedded in them and do things “that traditional fabrics cannot, including communicate, transform, conduct energy and even grow”. [33, 28]

2.2 Decision tree

This section contains some of the basic concepts of decision trees used for classification as described in [1], [2], and [34].

An advantage of using a decision tree technique for classification is the interpretability of the resulting model [34]. Figure 2.3 shows a simple decision tree that can be used to correctly classify the positive instances and negative instances, denoted by P and N respectively, developed from a training set shown in Figure 2.2.

There are two types of nodes in a decision tree: branch nodes and leaf nodes [34]. In Figure 2.3 the ‘outlook’, ‘humidity’, and ‘windy’ nodes are branch nodes, and ‘N’ and ‘P’ are leaf nodes [1]. The branch node at the top, in this case ‘outlook’ node, is called the root node and represents the entire feature space. Branch nodes below the root node, the children nodes, represent a disjoint subspace of their parent node’s entire feature space [34]. The paths from each branch node to its child nodes, the possible outcomes, represent classification rules that test the value of a specific attribute. For example, branch node ‘humidity’ is applicable to those instances where the attribute ‘outlook’ was tested and is equal to ‘sunny’, and branch node ‘windy’ has its classification rule applied only to the subspace where the attribute ‘outlook’ is ‘rain’. The values for the attributes in Figure 2.2 are discrete, such as the value of attribute ‘Windy’ could be either ‘true’ or ‘false’, and the value of attribute ‘Temperature’ could be ‘cold’, ‘mild’, or ‘hot’. A classification rule can handle either discrete, mutually exclusive values [1], such as this simple training set, or numeric values with inequality expressions that describe the decision boundaries [34, 2].

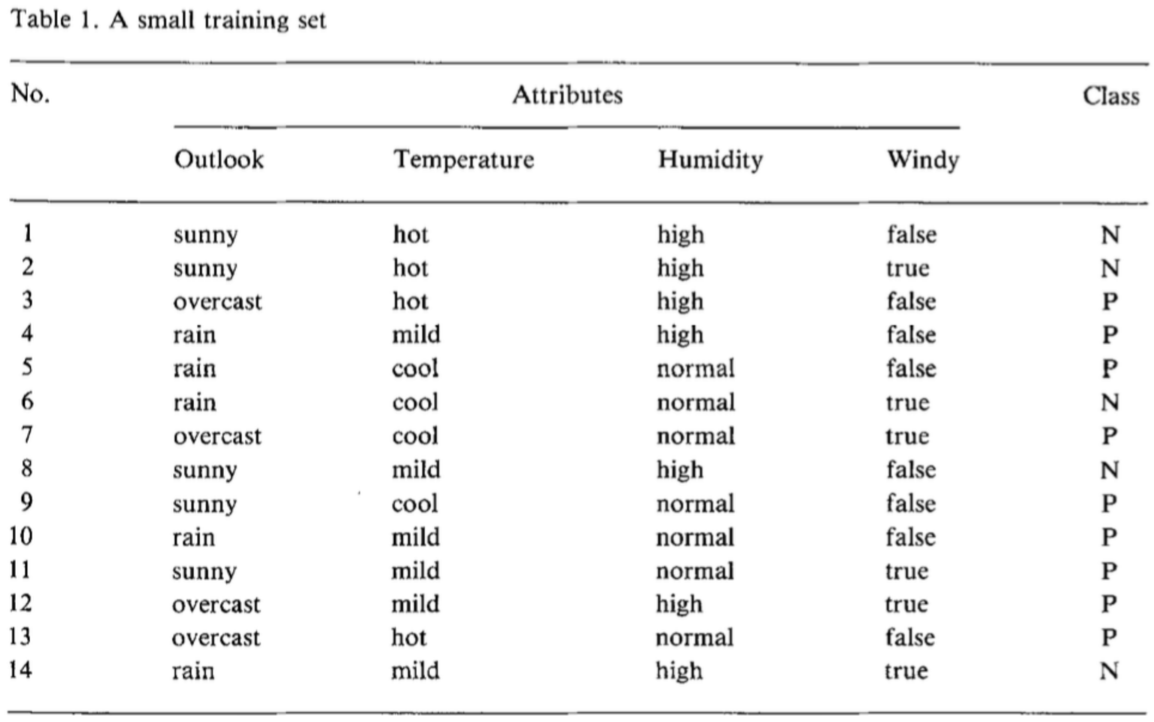

Table 1. A small training set

No.	Attributes				Class
	Outlook	Temperature	Humidity	Windy	
1	sunny	hot	high	false	N
2	sunny	hot	high	true	N
3	overcast	hot	high	false	P
4	rain	mild	high	false	P
5	rain	cool	normal	false	P
6	rain	cool	normal	true	N
7	overcast	cool	normal	true	P
8	sunny	mild	high	false	N
9	sunny	cool	normal	false	P
10	rain	mild	normal	false	P
11	sunny	mild	normal	true	P
12	overcast	mild	high	true	P
13	overcast	hot	normal	false	P
14	rain	mild	high	true	N

Figure 2.2: A simple decision tree training set [1]

The objective of a decision tree is to determine classification rules that can predict the class of any instance, or row in the case of Figure 2.2, from the value of its attributes and be expressed as a decision tree [1]. The attributes shown in the training set, Figure 2.2, are ‘Outlook’, ‘Temperature’, ‘Humidity’, and ‘Windy’, and there are 14 instances (rows) provided to train and test with. It is important to consider if the training set and its attributes are sufficient for developing the classification rules to predict accurately. If two instances have identical values for each attribute, although

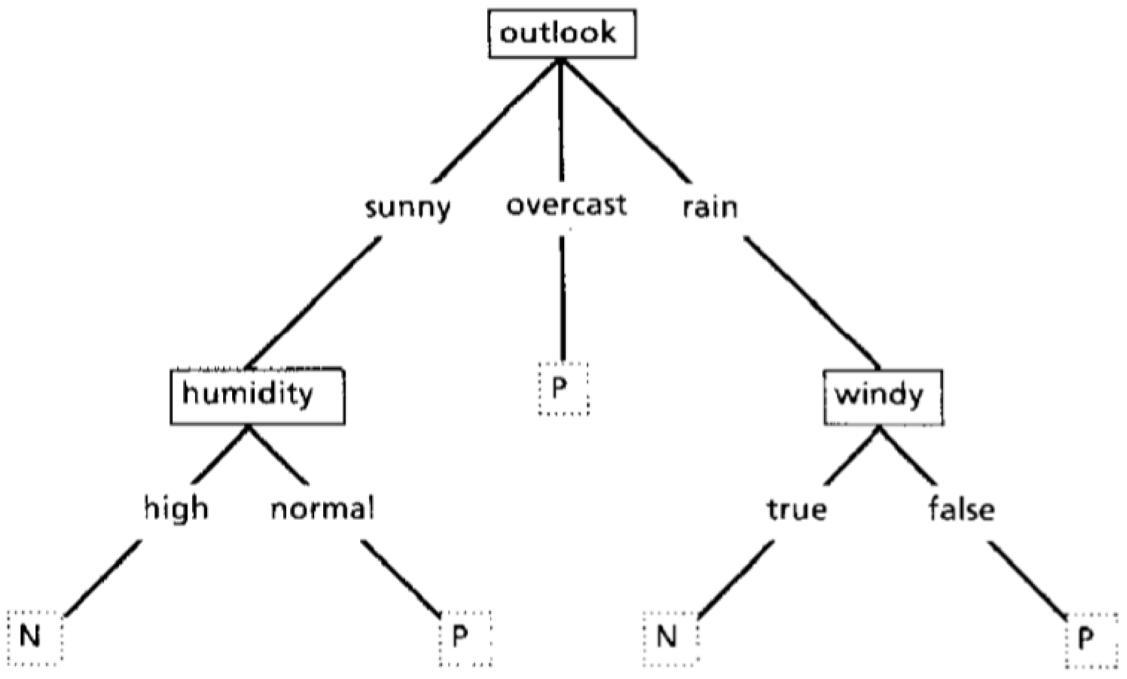

Figure 2.3: A simple decision tree for training set 2.2 [1]

they belong to different classes, this indicates that the training set is inadequate [1]. While this simple training set does not have any conflicting instances, it is critical that you consider the quality of your training set to ensure it is not inadequate.

Classification and Regression Trees (CARTs) are created by recursively partitioning – also called splitting – instances into groups with similar attribute values [2]. The possible combination of rules includes all partitions that can be obtained from the process of recursively splitting [2]. Multiple splits on the same attribute is also allowed [2]. Impurity reduction is used to determine the optimal attribute for splitting and the corresponding cut-point value [2]. After splitting with the optimal attribute and cut-point value the child nodes are more pure than the parent node [2]. The impurity reduction is measured by the difference between the impurity in the parent node and the average impurity in the child nodes [2]. The impurity in each node is calculated with Entropy measures, such as the Gini Index or the Shannon Entropy measures [2].

The stopping criteria for the recursive decision tree algorithm is commonly a threshold for the minimum number of remaining instances in a node's partition or a threshold for the minimum difference in impurity calculated [2]. To mitigate overfitting, tree pruning techniques [34, 2] and statistical stopping criteria [2] have been used. A classification model that performs well when tested on the training data and then performs poorly on unseen data is said to be overfitting the training data [2, 34]. Furthermore, a simpler decision tree is often preferable because it is more likely to have generalized the problem and will adapt well to predicting for new test data [1]. Figure 2.4 shows a more complex decision tree, in contrast to the simpler decision tree shown in Figure 2.3, that is equally as accurate on the same training set shown in Figure 2.2 [1]. Pruning a decision tree after it has been grown means removing the branches that do not contribute to improving the prediction accuracy when testing with cross-validation [2].

A disadvantage of recursive decision tree models is the instability with respect to small changes in training data [2]. If the first split and/or cut-point was chosen differently because of a small change in input training data, then the entire decision tree could be changed [2]. Figure 2.5 illustrates the high variability in decision trees with small changes in input training data [2].

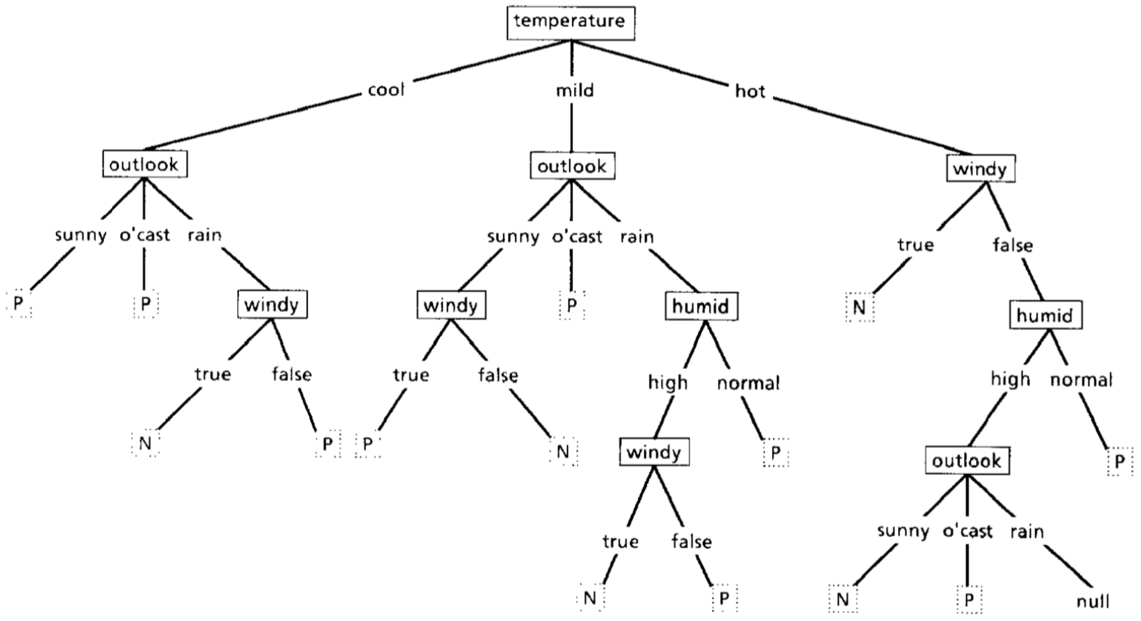

Figure 2.4: A complex decision tree for training set 2.2 [1]

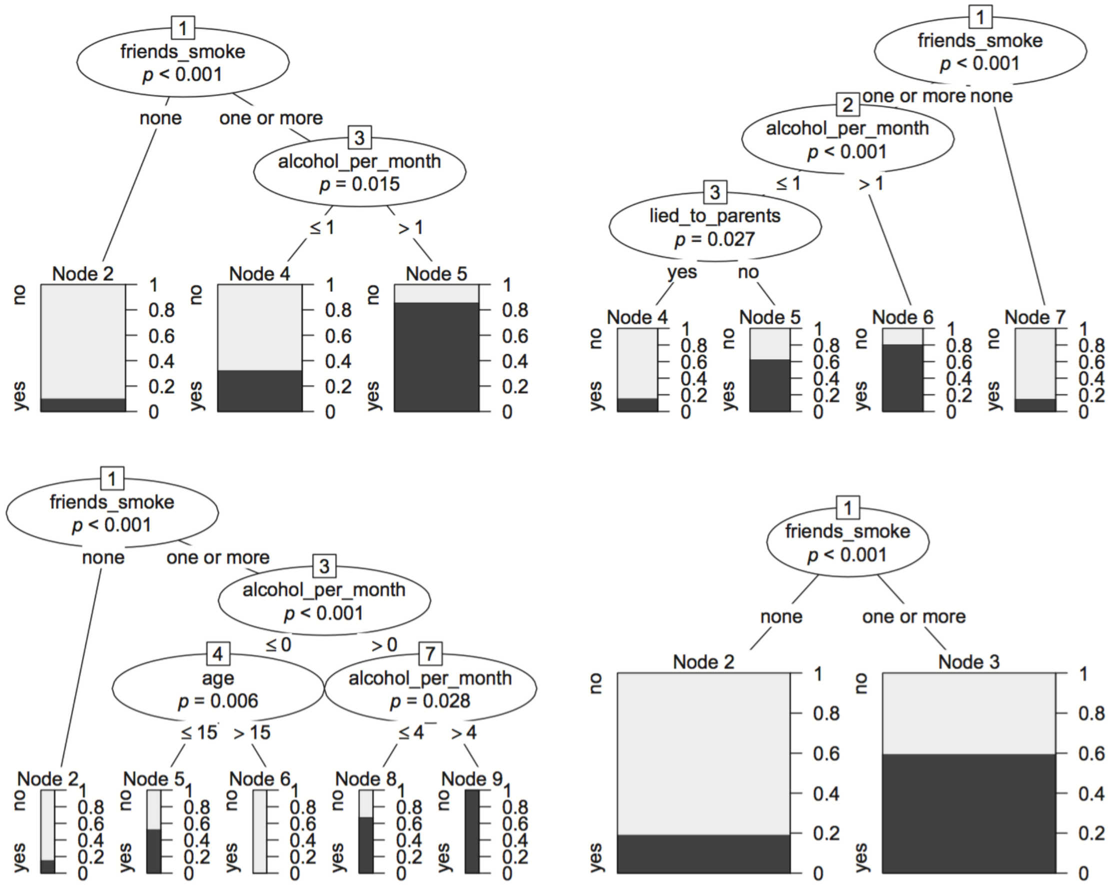

Figure 2.5: Four different decision trees generated with only small changes in training data [2]

2.3 Random forest

This section contains some of the basic concepts of random forest used for classification as described in [2], [34], and [35].

The random forest technique was proposed by Breiman (2001) and leverages multiple decision trees to predict an outcome [35]. Its output is determined by the prediction that appears the most often, the mode, of each of the individual decision trees [35, 2]. This technique is considered an 'ensemble learning' technique because it uses multiple models, in this case multiple decision trees [35]. In comparison to using a single decision tree, ensemble methods improve prediction accuracy [34, 2, 35], with the tradeoff being in their loss of interpretability of the resulting model [34].

Multiple trees, or an ensemble of trees, can be used to mitigate the instability of a single decision tree [2]. An ensemble of trees is created with random samples picked from the input training data [2, 34]. The instances excluded with each random sample can be considered out-of-bag and used as test samples for measuring out-of-bag prediction accuracy, which is more realistic and conservative [2]. In random forests, the attributes used for generating the tree are restricted and decided by variable selection, where a subset of attributes is randomly chosen from all of the attributes available in the data [2]. In contrast, the bagging [2, 34] method (not covered within the scope of this review) performs the same variable selection process as generating a single decision tree [2] along with similarly choosing random samples from the training data [2, 34]. The added restriction, in random forest over-bagging, of randomly selecting a subset of attributes for each split, helps to produce more diverse trees in the ensemble [2]. See Figure 2.6 for an example of a random forest ensemble of classification trees.

The three steps of a conventional random forest algorithm as described by Fernndez-Blanco [35] are:

1. Get n random samples from the original training dataset to use them as tree seeds.

2. For each tree seed, grow a non-pruned tree, and for each node, randomly choose m attributes, also called predictors, and determine the best cut-point to split with for those attributes.

3. Evaluate the different decision trees and the output is the mode of the predicted outcomes from each of the individual decision trees in the ensemble.

The Random forest technique provides a solution to overfitting experienced by single decision trees [35]. Using the fact that classification decision trees are, on average, correct with their predictions, known as 'unbiased predictors', [2], tree ensembles minimize overfitting with a set of diverse trees that tend to converge when the set is sufficiently large [35]. Randomly restricting the attributes to split and generate a tree with allows attributes that would otherwise not be chosen, and consequently discovers cross-attribute correlations and patterns that otherwise would have been missed [2]. An ensemble of trees provides an opportunity for less-likely attributes to be used with different attributes and produces a complex change on the predicted outcome [2]. The random restricting of attributes to split creates locally suboptimal splits that often improve the global prediction accuracy of the overall ensemble of decision trees [2].

Random forests have been shown to perform as well as Classification and Regression Tree (CART) single tree predictors [35]. This makes sense, given that when the decision trees are being grown the search is only performed on a smaller, random sample of data and there is no need for pruning the trees [35].

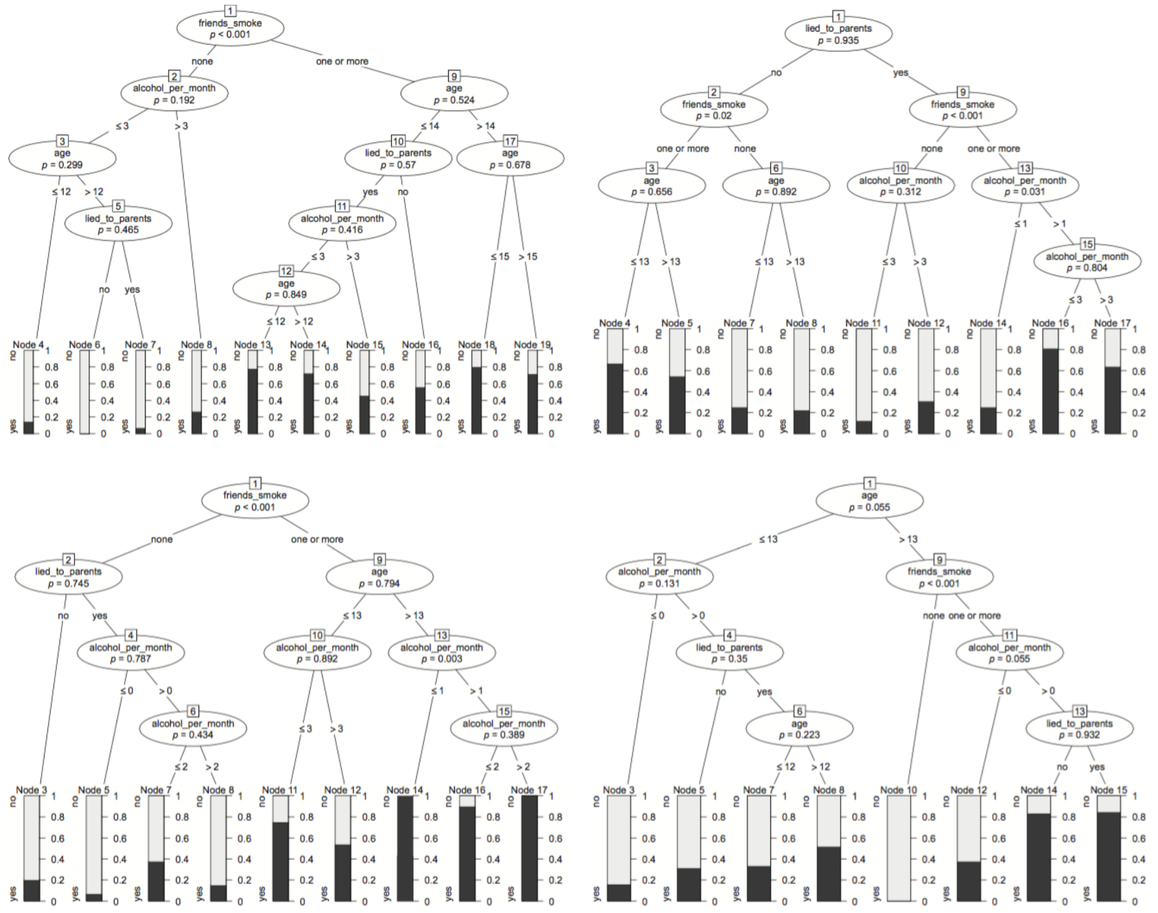
friends_smoke $p < 0.001$
alcohol_per_month $p = 0.192$
age $p = 0.524$
age $p = 0.299$
lied_to_parents $p = 0.57$
age $p = 0.678$
lied_to_parents $p = 0.465$
alcohol_per_month $p = 0.416$
age $p = 0.849$
lied_to_parents $p = 0.935$
friends_smoke $p = 0.02$
friends_smoke $p < 0.001$
age $p = 0.656$
age $p = 0.892$
alcohol_per_month $p = 0.312$
alcohol_per_month $p = 0.031$
alcohol_per_month $p = 0.804$
friends_smoke $p < 0.001$
lied_to_parents $p = 0.745$
age $p = 0.794$
alcohol_per_month $p = 0.787$
alcohol_per_month $p = 0.892$
alcohol_per_month $p = 0.003$
alcohol_per_month $p = 0.434$
alcohol_per_month $p = 0.389$
age $p = 0.055$
alcohol_per_month $p = 0.131$
friends_smoke $p < 0.001$
lied_to_parents $p = 0.35$
alcohol_per_month $p = 0.055$
age $p = 0.223$
lied_to_parents $p = 0.932$
none
one or more
yes
no

Figure 2.6: Example of a random forest ensemble of classification trees based on four random samples of smoking data (grown without stopping or pruning and with two attributes randomly selected in each split) [2]

2.3.1 Variable Importance

As you can see in Figure 2.6 interpreting the resulting model generated by random forests algorithm is not as simple as it would be with a single classification tree. The ensembles of trees in random forests are not nested and their attributes can appear in different positions throughout different trees, if they appear at all [2]. Variable importance is computed to quantify the relevance of each attribute, also referred to as a predictor variable, contributing to the overall ensemble of trees [2]. Figure 2.7 shows the variable importance scores for the predictor variables of smoking data [2].

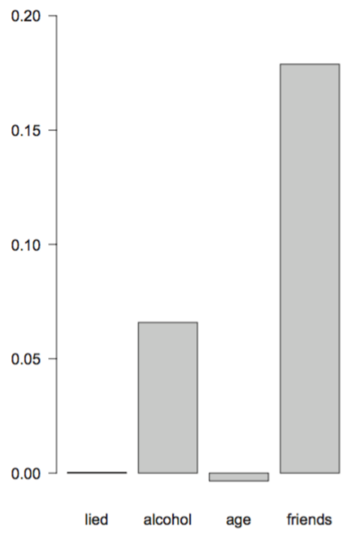

Figure 2.7: Example of permutation variable importance scores for the predictor variables of the smoking data from random forests [2]

The permutation accuracy importance measure is the most advanced variable importance method [2] and the measure that was used in this paper. The measure for variable importance is computed as the difference between the out-of-bag prediction accuracy before and after randomly permuting a variable and then averaged over all of the trees [2]. For example, permuting the ‘temperature’ variable from Figure 2.2

would mean randomly changing the value of ‘temperature’ between ‘hot’, ‘mild’, and ‘cool’. Along with the variable importance being zero – indicating the variable is irrelevant – it can also be negative in cases where permuting the variable actually improves prediction accuracy unexpectedly [2]. In Figure 2.7 the ‘age’ predictor variable is fairly irrelevant and in using the permutation accuracy importance measure it appears to have a negative importance score.

2.4 Support vector machine

This section summarizes the basic concepts of support vector machine (SVM) used for classification as described in [3], [4], [36], [37], and [38].

Support vector machine (SVM), also referred to as a support-vector network, is used for binary classification [4, 36, 3, 37]. The attributes of the input training data are referred to as features. SVM works by first mapping the input data into a higher dimensional feature space. A simple 1-dimension to 2-dimension mapping, Φ, could be:

$$\Phi : (x) \rightarrow (x, x^2) \tag{2.1}$$

See Figure 2.8 for a graphical representation of mapping the input data into higher dimensional feature space.

Then, the SVM model works to produce an optimal hyperplane in the new high dimensional feature space. The hyperplane separates the data into two groups, representing the two classes of the input data. In two-dimensional space we can separate instances into two groups with a line. In higher dimensional space we use hyperplanes.

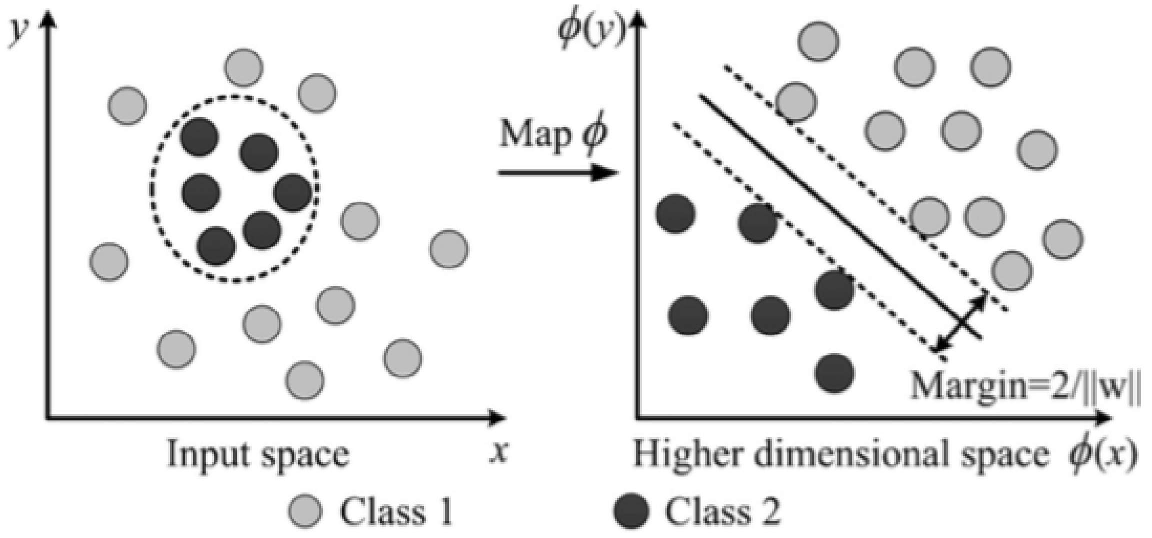

Figure 2.8: Graphical representation of mapping the input data into higher dimensional feature space [3]

An optimal hyperplane maximizes the margin or separation between the two groups. See Figure 2.9 for an example optimal hyperplane separating a 2-dimensional space. The instances on the margin of the hyperplane, marked with grey squares, are called the support vectors.

An advantage of SVM is the use of the kernel function [38]. The kernel function provides flexibility to SVM because it is not required to be linear [38] and you can even develop a custom kernel function specifically for the problem with human expertise. A kernel function is applied to pairs of vectors in the input space and acts as a distance measure, mapping data into their inner products with a simpler, one-dimensional feature space. Use of a kernel function is often referred to as a 'kernel trick' [36] because using inner product kernels in the input space allows classification without the computation cost of mapping and computing in higher dimensional feature spaces [37].

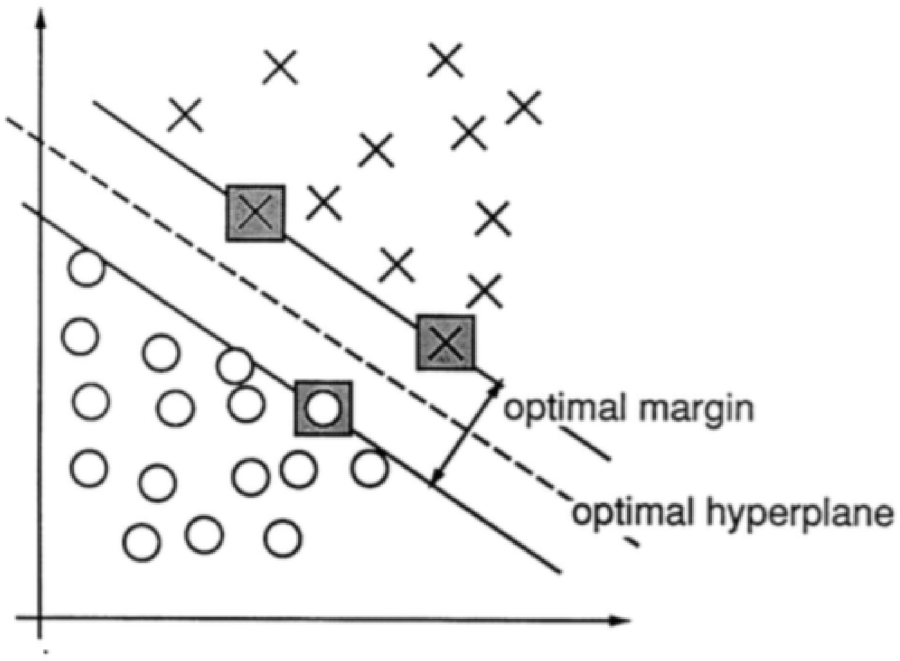

Figure 2.9: An example of a separable problem in a 2-dimensional space. The support vectors, marked with grey squares, define the margin of largest separation between the two classes. [4]

The linear kernel is used in this paper and defined as

$$K(x_i, x_{i'}) = (x_i \cdot x_{i'}) \tag{2.2}$$

for all pairs of vectors where $x_i \cdot x_{i'}$ is the dot-product of vectors x_i and $x_{i'}$.

The equation for an optimal hyperplane in the feature space is

$$0 = w \cdot x + b \tag{2.3}$$

where $w \in R^n$ represents the weights, x is the feature vector in feature space, and $b \in R$ is the bias [4, 36]. The weights, w, can be written as a linear combination of support vectors,

$$w = \sum_{support\ vectors} a_i z_i \tag{2.4}$$

where a_i are weights and z_i are the support vectors [4]. Thus, the linear decision function $I(z)$ in the feature space using the linear kernel can be written as:

$$I(z) = sign(\sum_{support\ vectors} a_i z_i \cdot x + b), \tag{2.5}$$

where $z_i \cdot x$ is the dot-product between support vectors z_i and vector x in the feature space [4].

The dataset used in this paper has more than 2 classes. There are five classes of activities and three to four classes representing the persons. Since support vector machines are explicitly designed to classify into two groups, a specialized approach is required to handle multiple classes in the classification dataset, referred to as multiclass-classification. The implementation of SVM used in this paper used the 'one-against-one', or 'one-versus-one', approach for multiclass-classification [39]. For k classes there are $\frac{k(k-1)}{2}$ binary classifiers trained, and then a voting scheme decides the appropriate single class predicted [39, 36]. Each binary classifier is given a newly constructed training dataset, such that one class is considered the positive class and another class is considered the negative class [36]. Finally, each of the binary classifiers can vote on the single class that they predict is correct, and the class with the most votes is considered the combined prediction.

In the case where the training data cannot be separated into 2 groups without errors the hyperplane with the minimal number of errors is optimal [4]. This is called the soft margin hyperplane [4]. Fortunately, optimizing the hyperplane for the minimal number of errors is a convex programming problem [4, 38]. This means there will be a unique solution to the convex problem representing the optimal hyperplane, and is an advantage over Neural Networks, which have multiple solutions for local minima [38]. A disadvantage of SVM is the transparency of its results, since it may be a very high dimension [38].

2.5 Neural network

This section summarizes the basic concepts of artificial neural networks (ANNs), also referred to as simply 'neural networks', used for classification as described in [5].

Artificial neural networks, used for information processing, are inspired by the interactions within the biological nervous system of the human brain [5]. Figure 2.10 shows the interaction between two biological neurons. The neurons are connected and communicate to each other through *synapses*, *dendrites* and *axons*. The dendrites of a neuron act as a set of inputs while, in contrast, the axon acts to as the neuron's ouput. The synapse is a junction that bridges the communication between one neuron's output, the axon, and another neuron's input, the dendrites.

Neurons communicate by "firing" an electrical impulse, called an action potential. This electrical impulse travels from the cell body through their axon towards the synapse. Upon reaching the synapse the electrical impulse triggers a release of chemical neurotransmitters that travel to the next neuron. Synapse can either be excitatory or inhibitory which increases or decreases the probability of the next neuron firing, respectively. Each synapse determines the magnitude of the impulse on the next neuron with an associated strength, or weight value. As a result, each neuron computes a weighted sum of the inputs received from the other neurons. If the total weighted sum received by the neuron is greater than some threshold then the neuron will fire.

There are typically 10^{11} active neurons in the human brain, each with thousands of connections to other neurons. This results in the total number of synapses exceeding 10^{14} throughout the human brain. Biological neural networks allow for massive parallelism of information processing by distributing signals over a large number of synapses simultaneously.

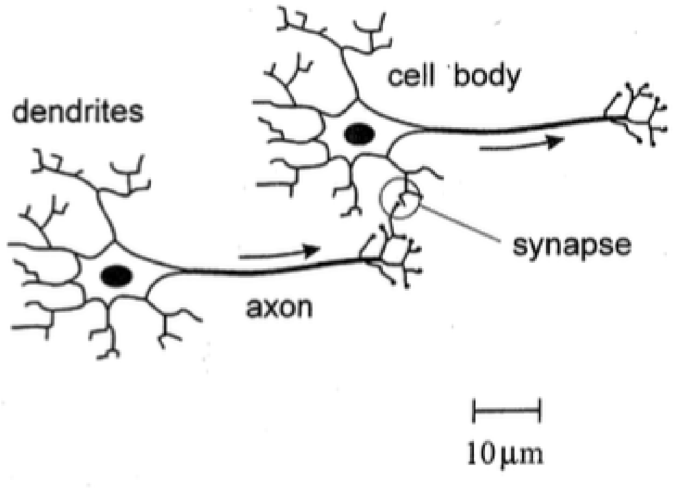

Figure 2.10: Simplified diagram of two biological neurons [5].

Both biological and artificial neural networks learn by changing their weight values, which is the strength of the synapses in biological networks [5]. For supervised learning, the backpropagation algorithm is used to quickly train artificial neural networks and adjust the weights to minimize the error between predicted and desired outputs [40, 5]. Figure 2.11 shows a model of an artificial neuron proposed by McCulloch and Pitts in 1943 [5]. Artificial neurons receive a set of d input variables

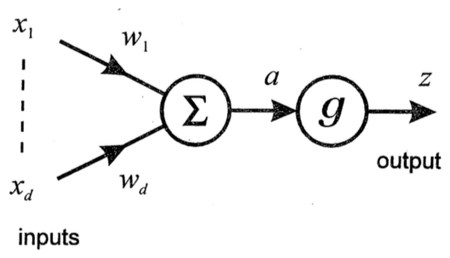

Figure 2.11: The McCulloch-Pitts model of a single neuron. $x_1, \ldots, x_d$ are inputs, $w_1, \ldots, w_d$ are weights, such that $a = \sum_i w_i x_i$ and $z = g(a)$ with a non-linear activation function $g()$ [5].

$x_1, \ldots, x_d$, which is analogous to the signals received from the dendrites in a biological neuron. The value x_i at input i is first multiplied by the respective weight w_i, which corresponds to the strength of the synapse in a biological network. The total input value a is the sum of the products of the input and weight values with the addition of an offset parameter w_0. Thus we can write the following equation for a:

$$a = \sum_{i=1}^{d} w_i x_i + w_0 \tag{2.6}$$

The offset parameter w_0 is called a *bias* and is similar to the firing threshold in a biological neuron. The output z of the artificial neuron is obtained by applying a non-linear *activation function* $g()$ to a giving the equation

$$z = g(a). \tag{2.7}$$

The value of the output z can be considered the average firing rate of a neuron. Figure 2.12 shows four typical activation functions. The linear activation function shown in Figure 2.12(a) is not very interesting and can be written simply as

$$g(a) = a. \tag{2.8}$$

The sigmoidal activation function shown in Figure 2.12(d) is used in most networks of practical interest, including this paper, and can be written as

$$g(a) = \frac{1}{1 + exp(-a)}. \tag{2.9}$$

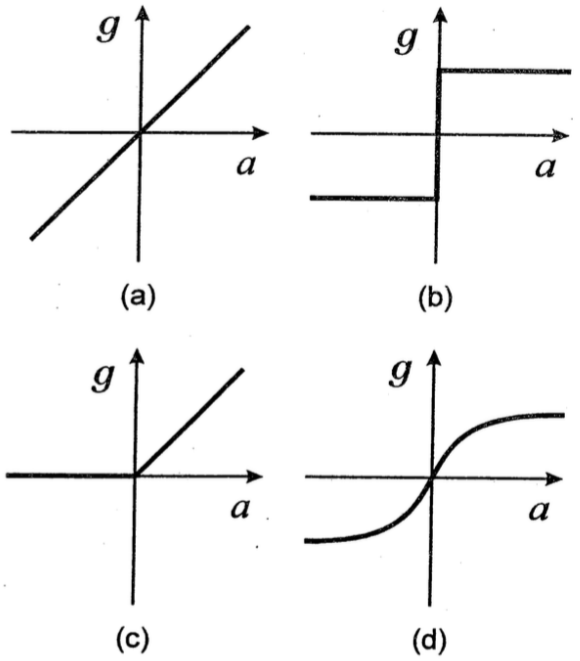

Figure 2.12: Four typical activation functions: (a) linear, (b) threshold, (c) threshold linear, (d) sigmoidal [5].

Artificial neural networks have been used for computing vision and speech recognition, including learning to recognize different characters from image pixels, known as optical character recognition (OCR), handwriting recognition, speech to text, and much more. Neural networks are strong candidates to solving problems which posses a subset of the following characteristics: (i) there is plenty of data to train with; (ii) it is difficult to provide a simple yet accurate model; (iii) new data needs to be processed quickly; (iv) the input data has modest levels of noise.

The advantages of being able to learn from modestly noisy input training data comes at the cost of requiring plenty of training data to begin with. Neural networks have the ability to learn a general solution to a problem, however this requires a sufficient training dataset that spans the possible input space. Another disadvantage is training the neural network is computationally intensive [5], especially given the large volume of input training data that is required. Fortunately, once the weight values have been trained in the neural network, new data can be processed very quickly and is ideal for processing a large volume of data or real-time applications.

2.6 K-means clustering

This section summarizes the basic concepts of K-means clustering algorithm as described in [41] and [42].

The K-means algorithm is given the number of clusters k to find, which also determines the number of centroids used [41]. A centroid is the average position of all the points or the middle of the cluster. The objective of the K-means algorithm for clustering is to start with random centroids, that are effectively initial guesses, and then determine the optimal and stable centroids for the clusters through subsequent iterations.

The k initial centroids can be selected randomly from the dataset or randomly created locations within the input space. Each of the input data instances are then assigned to the closest centroid. The closest centroid is determined by a distance metric, which is commonly the Euclidian distance. After each instance has been assigned to a centroid, a new centroid is calculated for each cluster. A new centroid is calculated as the average of all of the instances assigned to it.

This process is repeated and the entire dataset are reassigned to the new centroids. The algorithm will repeat until the cluster centroids do not change with subsequent iterations [41]. The k clusters can then represented as the final centroids.

An advantage of the K-means algorithm is its simplicity, and it is consequently easy to understand and implement [42]. However, a disadvantage of using K-means for clustering is the number of clusters needs to be provided in advance [42]. Fortunately, in this paper the number of clusters was known and equal to number of persons in the classification dataset.

2.7 Rough set theory

This section contains some of the basic concepts of rough set theory proposed by Pawlak [43] as described in [6]. The basic concept of rough set theory is representing a set as both lower and upper approximations instead of the traditional non-overlapping sets. A couple of disadvantages of the conventional non-overlapping sets are: Boundaries of sets or clusters are not always clearly defined, and objects may be equidistant from the center of multiple clusters. These are not weaknesses of rough sets, and in contrast rough sets are more flexible because they allow overlapping clusters and also less descriptive (specific) than fuzzy sets.

Let U denote a non-empty set of finite elements, also known as the universe. Let $R \subseteq U \times U$ be an equivalence relation on U. The set U is then partitioned into disjoint subsets called the equivalence classes, denoted by $U/R = (E_1, E_2, ..., E_n)$, where E_i is said to be an equivalence class of R. The equivalence classes are important to determine which elements are indistinguishable, or equivalent with respect to the equivalence relation R, from each other. Elements are considered indistinguishable from each other if they belong to the same equivalence class.

For example, let $U = \{a, b, c\}$ then

$$U \times U = \{(a,a),(a,b),(a,c),(b,a),(b,b),(b,c),(c,a),(c,b),(c,c)\} \tag{2.10}$$

such that

$$R \subseteq \{(a,a),(a,b),(a,c),(b,a),(b,b),(b,c),(c,a),(c,b),(c,c)\}. \tag{2.11}$$

A possible equivalence relation on U could then be

$$R = \{(a,a),(b,b),(b,c),(c,b),(c,c)\}. \tag{2.12}$$

The respective equivalence classes would be

$$U/R = \{\{a\}, \{b, c\}\}, \tag{2.13}$$

meaning that element a is only indistinguishable with itself, and elements b and c are indistinguishable from each other.

The approximation space is denoted by the pair $A = (U, R)$. The equivalence classes, U/R, are called the elementary or atomic sets in the approximation space, A. A composed set in A is the union of multiple elementary sets, with the empty set being considered a special composed set.

For an arbitrary set $X \subseteq U$ it may not be possible to determine a precise representation with elementary sets in A. Rough sets account for this by using both lower and upper approximations to represent the set, X.

The lower approximation, denoted by $\underline{A}(X)$, is a set comprised of only elements that definitely belong to the subset X, and is also known as the positive region. The elements that definitely belong to the subset X are determined by the composed set with all elementary sets which are subsets of X.

$$\underline{A}(X) = \{E_i | E_i \subseteq X\} \tag{2.14}$$

The upper approximation, denoted by $\overline{A}(X)$, is a set comprised of elements that both definitely and possibly belong to the subset X. The elements that possibly belong to the subset X are determined by the composed set with all of the elementary sets that have a non-empty intersection with X.

$$\overline{A}(X) = \{E_i | E_i \cap X \neq \emptyset\} \tag{2.15}$$

Note that $\underline{A}(X) \subseteq \overline{A}(X)$, or in other words, the lower approximation is a subset of the upper approximation for X.

The negative region is defined as elements that definitely do not belong to X and is expressed as the difference between the universe and the upper approximation, or $U - \overline{A}(X)$. The region that is covered by the upper approximation and not covered by the lower approximation is known as the boundary region, or $\overline{A}(X) - \underline{A}(X)$.

Figure 2.13 illustrates the lower (positive region) and upper approximation and negative region within the universe.

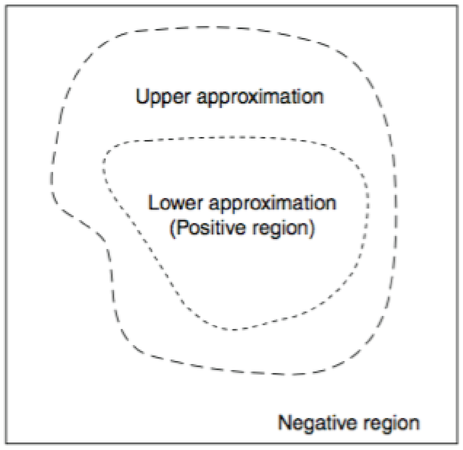

Figure 2.13: Rough set approximations [6]

The final representation of an ordinary set of X is the pair of lower and upper approximations, expressed as $[\underline{A}(X), \overline{A}(X)]$.

For example, consider $U = \{a, b, c, d\}$ and the equivalence classes

$$U/R = \{\{a\}, \{b, c\}, \{d\}\} \tag{2.16}$$

and set $X = \{a, b\}$. Note the element c is not contained in X, thus the elementary class $\{b, c\}$ is not a subset of X, or $\{b, c\} \nsubseteq X$. However, the elementary class containing a is $\{a\}$ is a subset of X, or $\{a\} \subseteq X$. Thus, the lower approximation is determined to be $\underline{A}(X) = \{a\}$.

We create a composed (union) set of all equivalence classes, denoted by $E_i \subseteq U/R$, where $E_i \cap X \neq \emptyset$ is true. The equivalence class $E_i = \{a\}$ definitely belongs to X, since a is in the lower approximation, $\underline{A}(X)$. However, we can verify this again with the equation $\{a\} \cap X = \{a\} \neq \emptyset$. For $E_i = \{b, c\}$ we determine $\{b, c\}$ is also possibly in X by $\{b, c\} \cap X = \{b\} \neq \emptyset$. We determine that $\{d\}$ is not in the upper approximation of X by $\{d\} \cap X = \emptyset$. Thus, the upper approximation is determined to be $\overline{A}(X) = \{a, b, c\}$.

The resulting representation of an ordinary set of X is

$$[\underline{A}(X), \overline{A}(X)] = [\{a\}, \{a, b, c\}]. \tag{2.17}$$

The negative region is

$$U - \overline{A}(X) = \{a, b, c, d\} - \{a, b, c\} = \{d\}, \tag{2.18}$$

and the boundary region is

$$\overline{A}(X) - \underline{A}(X) = \{a, b, c\} - \{a\} = \{b, c\}. \tag{2.19}$$

2.8 Adaptation of rough set theory for clustering

Let U be a set of objects. Rough sets were originally proposed using equivalence relations on U. However, it is possible to define a pair of lower and upper bounds $\big(\underline{A}(C), \overline{A}(C)\big)$, or a rough set for every set $C \subseteq U$ as long as the properties specified by Pawlak [44, 43] are satisfied. Yao *et al.* [45] described various generalizations of rough sets by relaxing the assumptions of an underlying equivalence relation. Such a trend towards generalization is also evident in rough mereology proposed by Polkowski and Skowron [46], and the use of information granules in a distributed environment by Skowron and Stepaniuk [47]. The present study uses a generalized view of rough sets. If one adopts a more restrictive view of rough set theory, the rough sets developed in this paper may have to be looked upon as interval sets [48].

Let us consider a hypothetical clustering scheme

$$U/P = \{C_1, C_2, \dots, C_k\} \tag{2.20}$$

that partitions the set U based on an equivalence relation P. Let us assume that due to insufficient knowledge it is not possible to precisely describe the sets $C_i, 1 \leq i \leq k,$

in the partition. However, it is possible to define each set $C_i \in U/P$ using its lower $\underline{A}(C_i)$ and upper $\overline{A}(C_i)$ bounds based on the available information. We will use vector representations, $\vec{u}, \vec{v}$ for objects and $\vec{c_i}$ for cluster C_i.

We are considering the upper and lower bounds of only a few subsets of U. Therefore, it is not possible to verify all the properties of the rough sets [44, 43]. However, the family of upper and lower bounds of $\vec{c_i} \in U/P$ are required to follow some of the basic rough set properties, such as:

(P1) An object $\vec{v}$ can be part of at most one lower bound

(P2) $\vec{v} \in \underline{A}(\vec{c_i}) \Longrightarrow \vec{v} \in \overline{A}(\vec{c_i})$

(P3) An object $\vec{v}$ is not part of any lower bound

$$\Updownarrow$$

$\vec{v}$ belongs to two or more upper bounds.

Property (P1) emphasizes the fact that a lower bound is included in a set. If two sets are mutually exclusive, their lower bounds should not overlap. Property (P2) confirms the fact that the lower bound is contained in the upper bound. Property (P3) is applicable to the objects in the boundary regions, which are defined as the differences between upper and lower bounds. The exact membership of objects in the boundary region is ambiguous. Therefore, property (P3) states that an object cannot belong to only a single boundary region. Note that (P1)-(P3) are not necessarily independent or complete. However, enumerating them will be helpful in understanding the rough

```
Genetic Algorithm:
    generate initial population, $G(0)$;
    evaluate $G(0)$;
    for(t = 1; solution is not found; t++)
        generate $G(t)$ using $G(t-1)$;
        evaluate $G(t)$;
```

Figure 2.14: Abstract view of a generational genetic algorithm

set adaptation of evolutionary, neural, and statistical clustering methods.

2.9 Genetic Algorithms

The origin of Genetic Algorithms (GAs) is attributed to Holland's [49] work on cellular automata. There has been significant interest in GAs over the last two decades. The range of applications of GAs includes such diverse areas as job shop scheduling, training neural nets, image feature extraction, and image feature identification [50]. This section contains some of the basic concepts of genetic algorithms as described in [50].

A genetic algorithm is a search process that follows the principles of evolution through natural selection. The domain knowledge is represented using a candidate solution called an *organism*. Typically, an organism is a single *genome* represented as a vector of length n:

$$c = (c_i \mid 1 \leq i \leq n), \tag{2.21}$$

where c_i is called a *gene*.

An abstract view of a generational GA is given in Fig. 2.14 and Fig. 2.15. A group of organisms is called a *population*. Successive populations are called *generations*. A

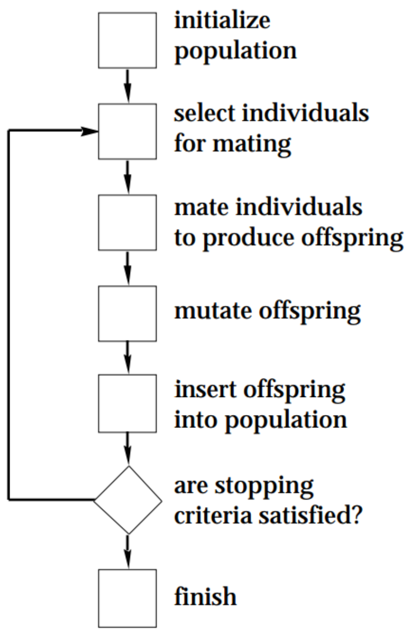
initialize population
select individuals for mating
mate individuals to produce offspring
mutate offspring
insert offspring into population
are stopping criteria satisfied?
finish

Figure 2.15: Flowchart of a generational genetic algorithm [7]

generational GA starts from initial generation $G(0)$, and for each generation $G(t)$, generates a new generation $G(t + 1)$ using genetic operators such as *mutation* and *crossover*. The mutation operator creates new genomes by changing values of one or more genes at random. The crossover operator joins segments of two or more genomes to generate a new genome.

2.10 Genetic Algorithms for Rough K-Medoid Clustering

This section describes a variation of the evolutionary rough K-means approach proposed by Lingras [12]. The proposal replaces K-means with K-medoids. A medoid is the most centrally located object in a given cluster. For k clusters, we will have k medoids. A genetic algorithm can be used to search for the most appropriate k

medoids. The genome will contain k genes, each corresponding to a medoid. This reduces the size of a genome from $k \times m$ by a factor of m to k. The smaller genomes will reduce the space requirements and facilitate faster convergence. The genes in the rough K-means algorithm were continuous real variables with no restriction on their values. The values of genes for the medoids will be discrete and limited to the number of objects in the dataset. If we number the objects from 1 to n, then each gene can take an integer value in the range $1 \ldots n$. This restriction on the values of genes will further reduce the search space allowing for even faster convergence.

The next step in the development of a rough K-medoid algorithm is to assign an object to the lower and/or upper bound of one of the clusters. The process is similar to the rough K-means algorithm. The major difference is that we use medoids instead of centroids of the clusters.

Let us assume that the clusters are represented by k medoids: $\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_k$. For each object vector, $\vec{v}$, let $d(\vec{v}, \vec{c}_j)$ be the distance between itself and the medoid of cluster $\vec{c}_j$. Let

$$d(\vec{v}, \vec{c}_i) = \min_{1 \leq j \leq k} d(\vec{v}, \vec{c}_j). \tag{2.22}$$

The ratios

$$\frac{d(\vec{v}, \vec{c}_i)}{d(\vec{v}, \vec{c}_j)}, \quad 1 \leq i, \quad j \leq k, \tag{2.23}$$

are used to determine the membership of $\vec{v}$. Let

$$T = \{j : \frac{d(\vec{v}, \vec{c}_i)}{d(\vec{v}, \vec{c}_j)} \leq \mathit{threshold} \text{ and } i \neq j\}. \tag{2.24}$$

1. If $T \neq \emptyset$, $\vec{v} \in \overline{A}(\vec{c}_i)$ and $\vec{v} \in \overline{A}(\vec{c}_j), \forall j \in T$. Furthermore, $\vec{v}$ is not part of any lower bound. The above criterion guarantees that property (P3) is satisfied.

2. Otherwise, if $T = \emptyset$, $\vec{v} \in \underline{A}(\vec{c}_i)$. In addition, by property (P2), $\vec{v} \in \overline{A}(\vec{c}_i)$.

It should be emphasized that the approximation space A is not defined based on any predefined relation on the set of objects. The lower and upper bounds are constructed based on the criteria described above.

The next step in calculating the fitness of a genome is to measure the validity of a clustering scheme. We will use one of the most intuitive distance-based validity measures. The measure will accumulate the distances of the objects assigned to a cluster and its medoid as determined by the GAs:

$$\triangle = \sum_{i\ =\ 1}^{k} \sum_{\vec{u} \in \vec{c}_i} d(\vec{u}, \vec{c}_i), \tag{2.25}$$

where the function d provides the distance between two vectors. The distance $d(\vec{u}, \vec{v})$ is given by:

$$d(\vec{u}, \vec{v}) \ = \ \sqrt{\frac{\sum_{j=1}^{m} (u_j - v_j)^2}{m}}. \tag{2.26}$$

We need to adapt the above measure for rough set theory by creating lower and upper versions of the error as:

$$\underline{\triangle} = \sum_{i\ =\ 1}^{k} \sum_{\vec{u} \in \underline{A}(\vec{c}_i)} d(\vec{u}, \vec{c}_i), \text{ and} \tag{2.27}$$

$$\overline{\triangle} = \sum_{i\,=\,1}^{k} \sum_{\vec{u} \in \overline{A}(\vec{c}_i) - \underline{A}(\vec{c}_i)} d(\vec{u}, \vec{c}_i). \tag{2.28}$$

The rough error is then calculated as a combination of the lower and upper error:

$$\triangle_{rough} = w_l \times \underline{\triangle} + w_u \times \overline{\triangle}. \tag{2.29}$$

The rough error described above is based on the distances between patterns. However, we know the categorization of the patterns based on activity and the person performing the activity. In this experiment, we will focus on the categorization of the data based on the person. We first need to make a correspondence between a cluster and the most predominant class in the upper bound of that cluster. We then count the number of correctly classified patterns. The error in classification will be the number of incorrectly classified patterns, $wrongClasses$. We then take a weighted combination of $\triangle_{rough}$ and $wrongClasses$:

$$objective = w_d \times \triangle_{rough} + w_c \times wrongClasses, \tag{2.30}$$

where w_d is the weight attached to the rough error and w_c is the weight attached to the classification error. Our GAs will minimize the objective functions given by Eq. 2.30.

Chapter 3

Study Data and Design of Experiments

3.1 Study Data

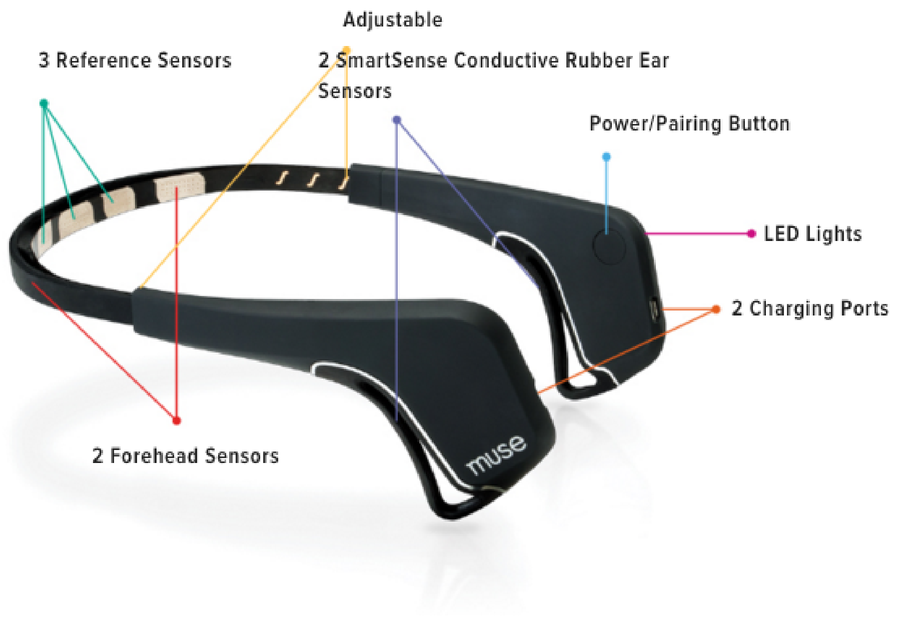

Figure 3.1: Labelled Muse headband

The Muse headband uses four sensors, two located on the forehead, and two behind the ears. Three additional sensors on the forehead are used as reference sensors by the device [25]. See Figure 3.1 for a labelled Muse headband. Although the Muse API also provides access to the signals for muscle movement and accelerometer data, our research focussed on the EEG data. Muse provides this data in a variety of formats.

At the lowest level, the analog microvolts signals are recorded by the four sensors which, by default, are sampled at a rate of 220 Hz. These EEG signals are then compressed in order to stream the data over Bluetooth. The signals are compressed via Golomb Encoding and further quantized to reduce their size. Full details on Muse's compression algorithm are available via their online manual [51]. Muse offers both absolute band powers and relative band powers. The absolute band power is computed as the logarithm of the sum of the power spectral data of the EEG over the specified frequency range (alpha, beta, delta, gamma, theta). The power spectral density is computed via Fast Fourier Transform on the device. The relative band powers normalize the absolute power bands as a percentage of the total absolute power bands [52], resulting in values between 0 and 1. This is calculated by:

$$s_r = \frac{10^{s_{abs}}}{10^{\alpha_{abs}} + 10^{\beta_{abs}} + 10^{\delta_{abs}} + 10^{\gamma_{abs}} + 10^{\theta_{abs}}} \tag{3.1}$$

where s_r is the relative frequency range (alpha, beta, delta, gamma or theta) being calculated and s_{abs} is the frequency range's absolute value.

It was assumed that the signals will be able to help us extract signature patterns for individual users as well as various activities. Five individuals participated in the original data collection programs. These individuals worked very closely with each other and used similar setups for data collection. Five activities that represent day-to-day functions performed by most people were identified as a proof of concept. These activities were as follows:

1. Reading: A user read a magazine for 1-3 minutes

2. Doing nothing: A user sat quietly for 1-3 minutes

3. Watching video: A user watched a video for 1-3 minutes

4. Game playing: A user played a computer game for 1-3 minutes

5. Listening to music: A user listened to music for 1-3 minutes

The above activities were repeated ten times for five individuals resulting in a total of fifty datasets.

3.2 Knowledge Representation

One of the key aspects of any data mining activity is representing real world entities using the pertinent numeric data available for them. In our case, we want to capture individuals and their activities using the signals emanating from their brain. As mentioned before, the Muse headband collects data from four positions around the head. Each position provides five types of waves: alpha, beta, gamma, delta, and theta. That means at any given point in time we receive a record with twenty values. There will be a stream of these twenty-valued records that will be recorded at a discrete time interval of 0.1 seconds. That means if we record activity for one person for one minute we will have a total of 600 records. Realistically, if we are recording an activity for a person, we cannot put an exact time limit on each person. In our experiment, the recording time per activity ranged anywhere from 60 to 180 seconds.

Tables 3.1 and 3.2 show the summary statistics for all the five waves: alpha, beta, gamma, delta, and theta for each of the four locations for all the participants. It can

Location	Wave	Min	25%	Median	75%	Max	Mean	Std. Dev
Front Right	Alpha	0.01	0.15	0.22	0.30	0.80	0.23	0.11
Front Right	Beta	0.01	0.09	0.14	0.20	0.68	0.15	0.09
Front Right	Delta	0.00	0.11	0.17	0.23	0.68	0.18	0.09
Front Right	Gamma	0.01	0.14	0.21	0.29	0.77	0.23	0.12
Front Right	Theta	0.00	0.09	0.15	0.21	0.59	0.16	0.08
Front Left	Alpha	0.01	0.09	0.16	0.24	0.62	0.17	0.10
Front Left	Beta	0.00	0.08	0.15	0.21	0.60	0.16	0.10
Front Left	Delta	0.00	0.09	0.15	0.22	0.62	0.16	0.10
Front Left	Gamma	0.01	0.21	0.32	0.47	0.95	0.35	0.17
Front Left	Theta	0.02	0.23	0.38	0.56	0.96	0.40	0.21
Back Right	Alpha	0.01	0.25	0.39	0.54	0.98	0.41	0.20
Back Right	Beta	0.00	0.20	0.32	0.47	0.95	0.34	0.18
Back Right	Delta	0.00	0.05	0.08	0.12	0.52	0.09	0.06
Back Right	Gamma	0.00	0.07	0.13	0.19	0.68	0.14	0.09
Back Right	Theta	0.00	0.05	0.09	0.15	0.69	0.11	0.08
Back Left	Alpha	0.00	0.04	0.08	0.13	0.62	0.10	0.08
Back Left	Beta	0.01	0.12	0.17	0.22	0.59	0.17	0.07
Back Left	Delta	0.01	0.09	0.13	0.18	0.58	0.14	0.07
Back Left	Gamma	0.01	0.10	0.15	0.20	0.52	0.15	0.07
Back Left	Theta	0.01	0.11	0.16	0.22	0.62	0.17	0.08

Table 3.1: Statistical summary of signals from all the channels - Used for the classification experiments

be seen that the values for each wave have similar ranges. However, the ranges for different waves vary considerably. For example, alpha values range from 9 Hz to 13 Hz, while beta values range from 12 Hz to 30 Hz [52]. In our dataset, using relative band powers with values between 0 and 1, the alpha values range from 0.00 to 0.98, while delta values range from 0.00 to 0.68. Therefore, we normalized the values for each wave using 90% of the maximum value for that wave. That made sure that the values for all the waves were in the same range.

Another issue with the data collection was the variable length of time for different activities as well as the length of each record. The length of the record could vary anywhere from 60 to 180 seconds. Figure 3.2 shows an example of one of the waves,

Location	Wave	Min	25%	Median	75%	Max	Mean	Std. Dev
Front Right	Alpha	0.01	0.16	0.24	0.32	0.80	0.25	0.12
Front Right	Beta	0.01	0.10	0.15	0.22	0.68	0.16	0.09
Front Right	Delta	0.00	0.12	0.18	0.25	0.68	0.19	0.09
Front Right	Gamma	0.01	0.15	0.23	0.32	0.77	0.25	0.13
Front Right	Theta	0.00	0.10	0.15	0.22	0.59	0.16	0.09
Front Left	Alpha	0.01	0.09	0.16	0.24	0.62	0.17	0.10
Front Left	Beta	0.00	0.11	0.17	0.22	0.60	0.17	0.09
Front Left	Delta	0.00	0.10	0.15	0.23	0.62	0.17	0.11
Front Left	Gamma	0.01	0.19	0.29	0.43	0.95	0.32	0.18
Front Left	Theta	0.02	0.21	0.36	0.55	0.96	0.39	0.21
Back Right	Alpha	0.01	0.23	0.35	0.53	0.98	0.38	0.19
Back Right	Beta	0.00	0.17	0.28	0.44	0.95	0.32	0.19
Back Right	Delta	0.00	0.05	0.09	0.13	0.46	0.09	0.06
Back Right	Gamma	0.00	0.07	0.12	0.18	0.64	0.14	0.09
Back Right	Theta	0.00	0.06	0.10	0.16	0.55	0.11	0.07
Back Left	Alpha	0.00	0.05	0.08	0.14	0.62	0.10	0.08
Back Left	Beta	0.01	0.11	0.16	0.21	0.59	0.17	0.07
Back Left	Delta	0.01	0.09	0.13	0.18	0.58	0.14	0.07
Back Left	Gamma	0.01	0.09	0.14	0.19	0.52	0.14	0.07
Back Left	Theta	0.01	0.10	0.14	0.20	0.62	0.15	0.08

Table 3.2: Statistical summary of signals from all the channels - Used for the clustering experiments

alpha, from position 3 while player 1 was playing a game for 60 seconds. In order to fix the length of the record to a fixed and more manageable value, we studied the frequency distribution of the records. After experimenting with different numbers of bins, we decided to use 8 bins with 5%, 15%, 25%, 50%, 75%, 85%, 95% values to represent the histogram. Here if x is 5% value, it means that 5% of the values will be less than or equal to x. Figure 3.3 shows the histogram of values for the wave shown in Figure 3.2. As we can see, the representation consists of only 8 values and captures the essence of each wave, the position, and the person in a concise and consistent manner. Since we have a total of four positions, five waves, and eight histogram bins, we have a total of $4 \times 5 \times 8 = 160$ values to represent each record of an activity. For

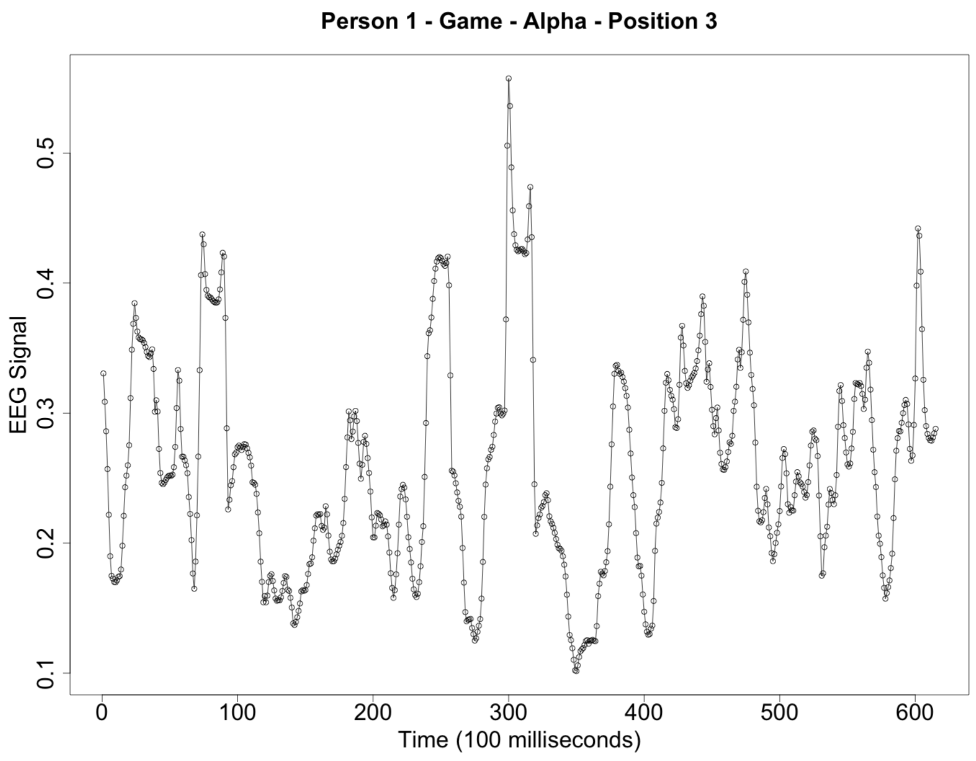

Figure 3.2: The alpha wave from position 3 for person 1 playing a game

five persons and five activities this process was repeated ten times, giving a total of $5 \times 5 \times 10 = 250$ records. We will use these records to train classification models to predict the person as well as the activity based on a recording from the Muse headband.

3.3 Experimental design

There were five individuals involved in the data collection. However, the number of records collected for each of the individuals varied. The individuals with a very small number of records tended to get neglected by the techniques in favour of individuals with a larger number of records. Therefore, in the initial experiments reported in this

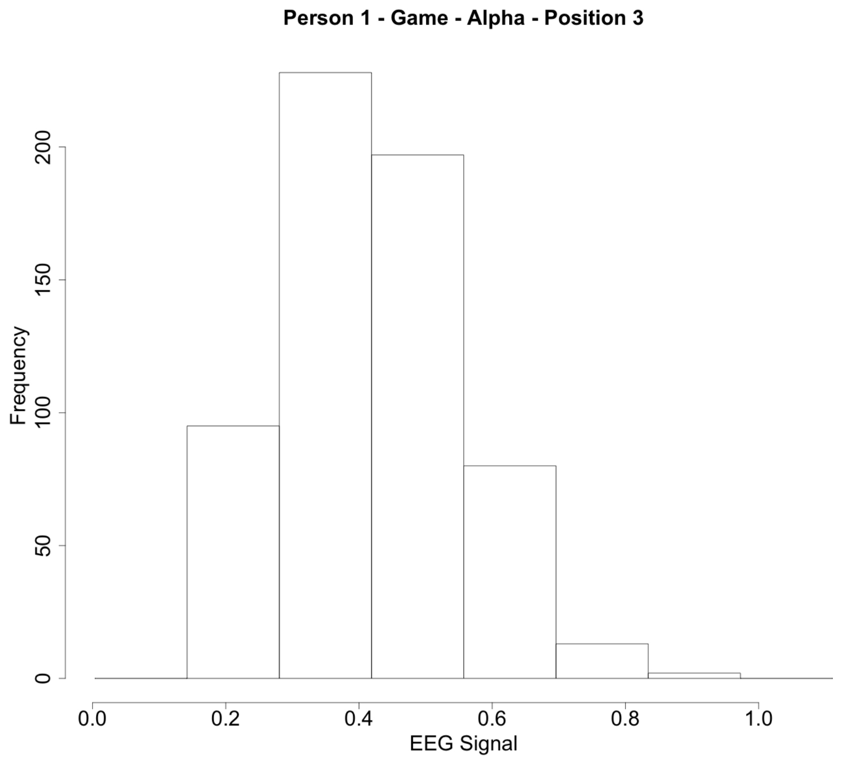

Figure 3.3: The histogram of the alpha wave from position 3 for person 1 playing a game

paper, we have used data from three and four of the original five individuals for the classification and clustering experiments, respectively.

3.3.1 Classification

In the initial classification experiments reported in this paper, we have used data from four individuals. Individual 0 had 23 records, individuals 1 and 2 had 50 records each, and individual 3 had 55 records, resulting in a total of 178 records.

The dataset used to develop the classification models therefore consists of 178 records. Each record contains 160 attributes. We used the following four well-known classifiers:

1. Decision tree
2. Random forests
3. Support vector machine (SVM)
4. Neural network

The decision tree classifier was used in eight experimental groups corresponding to eight different values $\{1, 2, 3, 4, 5, 10, 15, 20\}$ for the *minsplit* parameter, representing the minimum number of instances that must exist in a node in order for a split to be attempted. The *mtry* parameter – representing the number of randomly chosen attributes in the variable selection step – for the random forest classifier was set to 100. The linear kernel was used for the SVM classifier, as shown in Equation 2.5. The sigmoidal activation function was used for neural network classifier, as shown in Equation 2.9. The *maxit* parameter – representing the maximum number of iterations – of the neural network classifier was set to be 10, 25, 50, 100, or 1,000 for each of the predictions.

Each classifier was applied to three predictions:

1. Predicting a person - Four classes corresponding to four persons
2. Predicting an activity - Five classes corresponding to five activities
    - Reading
    - Doing nothing
    - Watching a video

- Playing a game
- Listening to music

3. Predicting a person as well as the activity - Twenty classes corresponding to the cross-product of the set of persons and the set of activities

To test how well our knowledge representation and classifiers describe the activities we first applied the classifiers to the entire dataset. This allowed us to study the importance of different attributes in the classification process.

In Tables 4.5, 4.6, and 4.7 you can see the most significant independent variables obtained by the random forest classifier when classifying by Person, Activity, or Person and Activity, respectively. The most significant independent variable for predicting a person, activity, or person and activity was bin 1 (0% to 5%) of the beta wave at location 1, bin 1 (0% to 5%) of the theta wave at location 2, and bin 7 (85% to 95%) of the theta wave at location 4, respectively. The theta wave at location 2 is significant for predicting the person or the activity.

In tables 4.8, 4.9, and 4.10 you can see the least ten significant independent variables obtained by the random forest classifier when classifying by Person, Activity, or Person and Activity, respectively. The least significant independent variable for predicting a person, activity, or person and activity was bin 8 (95% to 100%) of the alpha wave at location 4, bin 8 (95% to 100%) of the theta wave at location 4, and bin 8 (95% to 100%) of the theta wave at location 4, respectively. The signals from locations 3 and 4 are shown to be often the least significant variable overall for predicting person, activity, or person and activity. This could indicate that the back of the brain, the occipital lobe, is not as significant for differentiating the activities studied.

Using a complete dataset for training can lead to over training of the classifiers. The classifier may not work for new datasets. In order to see if the models were general enough to predict new recordings, we applied ten-fold cross-validation.

In ten-fold cross-validation [53] 10% of randomly selected data from the dataset is set aside for testing the model. The remaining 90% of the dataset is used for training the model. This process is repeated ten times and the results are summarized. From our tests we learned that our knowledge representation paired with SVM, random forest, or neural networks could predict a class for person or activity with fairly high precision. However, using a decision tree classifier proved to be ineffective.

The primary objective of this experiment was to explore the effectiveness of a number of well-known classification techniques combined with a data representation that reduces variable length signals to more manageable and uniformly fixed length distributions. Using this representation and the classification techniques, the hypothesis was that the EEG brain signals will be identifiable for both persons and activities with a reasonable degree of precision.

3.3.2 Semi-Supervised Evolutionary Learning

In the initial clustering and semi-supervised evolutionary learning experiments reported in this paper, we used data from three individuals. Individual 0 had 23 records, individuals 1 and 2 had 50 records each, resulting in a total of 123 records.

We used the two-point crossover technique for the genetic algorithm. The two-point crossover technique uses two points to divide both of the parents' genomes into three partial genomes. As shown in Fig. 3.4, the children genomes are generated by swapping the middle partial genome, between the two points, of the parents' genomes [54].

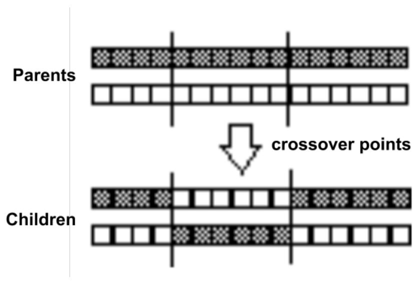

Figure 3.4: Two-Point Crossover for Genetic Algorithms [7]

The parameters for rough clustering were set as follows: $threshold = 1.1, w_u = 0.7, w_l = 0.3$. For multi-objective optimization, the weight for distance-based optimization w_d was set at 0.75, and the weight for classification w_c was equal to 0.25. Since the assignment of patterns to clusters is based on distance measure, the distance tended to influence the optimization more than the wrong classification. Therefore, the evolutionary semi-supervised crisp and rough clustering was run multiple times and solutions that provided the best classifications were chosen. These multiple runs also allowed us to steer out of locally optimal solutions.

The primary objective of this experiment was to explore the effectiveness of Euclidean distance between brain signals to identify an individual person. The hypothesis was that signals from an individual will be similar and belong to the same cluster. The hypothesis can be tested with the well-known K-means clustering algorithm.

The K-means clustering algorithm was selected because of its simplicity. Furthermore, the study explored the possibility of influencing the clustering with the known categorization using evolutionary crisp and rough K-medoid algorithms. We used confusion matrices for detailed analysis and the precision of clustering in identifying the individuals as two evaluation measures.

Chapter 4

Classification

Tables 4.1, 4.2, 4.3, and 4.4 and Figures 4.1, 4.2 4.3 show the precision [55, 56] of prediction for the four classifiers: decision tree, SVM, neural networks, and random forest, respectively. For each classifier, Tables 4.1, 4.2, 4.3, and 4.4 report the accuracy for predicting either the activity, person, or both. The results include training for the entire dataset as well as by ten-fold cross-validation.

Our tests from training on the entire dataset showed SVM and random forest were able to successfully predict a class for person, activity, or person and activity with 100% precision. Furthermore, neural networks were also able to predict person with 100% and activity with 95%, although this method did not perform well for predicting person and activity together. Overfitting was most prominent for predicting activity; however, the precision values remained fairly high after switching to using ten-fold cross-validation.

Precision tells us the likelihood of the classifier being correct when predicting a class. Precision can be calculated by dividing the diagonal value in a row by the sum of the rows of a confusion matrix [57]. The precision values for ten-fold cross-validation are all above 92% for predicting the person with SVM, neural networks, or random forest. By comparison, the precision values in Table 4.1 show that decision

	Classifier	Prediction Variable	Precision	
			Entire Dataset	Ten-Fold Cross Validation
1	Decision Tree (minsplit: 1)	Person	71.35	64.1
2	Decision Tree (minsplit: 1)	Activity	23.03	23.1
3	Decision Tree (minsplit: 1)	Person+Activity	6.18	5.65
4	Decision Tree (minsplit: 2)	Person	71.35	65.2
5	Decision Tree (minsplit: 2)	Activity	23.03	23.05
6	Decision Tree (minsplit: 2)	Person+Activity	6.18	5.64
7	Decision Tree (minsplit: 3)	Person	71.35	59.99
8	Decision Tree (minsplit: 3)	Activity	23.03	23.05
9	Decision Tree (minsplit: 3)	Person+Activity	6.18	5.65
10	Decision Tree (minsplit: 4)	Person	71.35	63.51
11	Decision Tree (minsplit: 4)	Activity	23.03	23.07
12	Decision Tree (minsplit: 4)	Person+Activity	6.18	5.64
13	Decision Tree (minsplit: 5)	Person	71.35	65.2
14	Decision Tree (minsplit: 5)	Activity	23.03	23.05
15	Decision Tree (minsplit: 5)	Person+Activity	6.18	5.65
16	Decision Tree (minsplit: 10)	Person	71.35	62.33
17	Decision Tree (minsplit: 10)	Activity	23.03	23.02
18	Decision Tree (minsplit: 10)	Person+Activity	6.18	5.64
19	Decision Tree (minsplit: 15)	Person	71.35	61.62
20	Decision Tree (minsplit: 15)	Activity	23.03	23.11
21	Decision Tree (minsplit: 15)	Person+Activity	6.18	5.65
22	Decision Tree (minsplit: 20)	Person	71.35	62.37
23	Decision Tree (minsplit: 20)	Activity	23.03	23.1
24	Decision Tree (minsplit: 20)	Person+Activity	6.18	5.63

Table 4.1: Prediction accuracy of Decision Tree classifiers

	Classifier	Prediction Variable	Precision	
			Entire Dataset	Ten-Fold Cross Validation
25	Support Vector Machine (Linear)	Person	100	95.39
26	Support Vector Machine (Linear)	Activity	100	77.51
27	Support Vector Machine (Linear)	Person+Activity	100	84.97

Table 4.2: Prediction accuracy of Support Vector Machine classifiers

	Classifier	Prediction Variable	Precision	
			Entire Dataset	Ten-Fold Cross Validation
28	Neural Network (maxit: 10)	Person	43.82	55.07
29	Neural Network (maxit: 10)	Activity	28.09	33.28
30	Neural Network (maxit: 10)	Person+Activity	12.36	10.58
31	Neural Network (maxit: 25)	Person	40.45	64.57
32	Neural Network (maxit: 25)	Activity	41.01	33.34
33	Neural Network (maxit: 25)	Person+Activity	11.24	14.72
34	Neural Network (maxit: 50)	Person	57.87	71.96
35	Neural Network (maxit: 50)	Activity	34.27	37.11
36	Neural Network (maxit: 50)	Person+Activity	15.73	16.26
37	Neural Network (maxit: 100)	Person	80.34	82.62
38	Neural Network (maxit: 100)	Activity	46.63	40
39	Neural Network (maxit: 100)	Person+Activity	17.98	25.93
40	Neural Network (maxit: 1000)	Person	100	94.46
41	Neural Network (maxit: 1000)	Activity	94.94	69.77
42	Neural Network (maxit: 1000)	Person+Activity	24.16	51.33

Table 4.3: Prediction accuracy of Neural Network classifiers

	Classifier	Prediction Variable	Precision	
			Entire Dataset	Ten-Fold Cross Validation
43	Random Forest	Person	100	92.85
44	Random Forest	Activity	100	75.78
45	Random Forest	PersonAndActivity	100	77.71

Table 4.4: Prediction accuracy of Random Forest classifiers

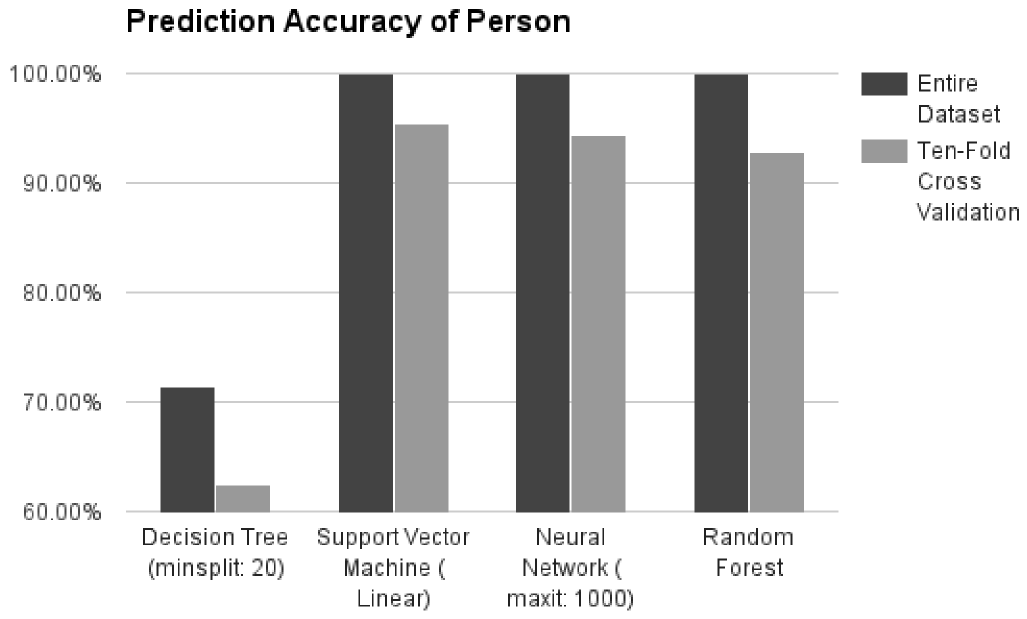

Figure 4.1: Comparison of prediction accuracy of person for decision tree, SVM, neural network, random forest classifiers

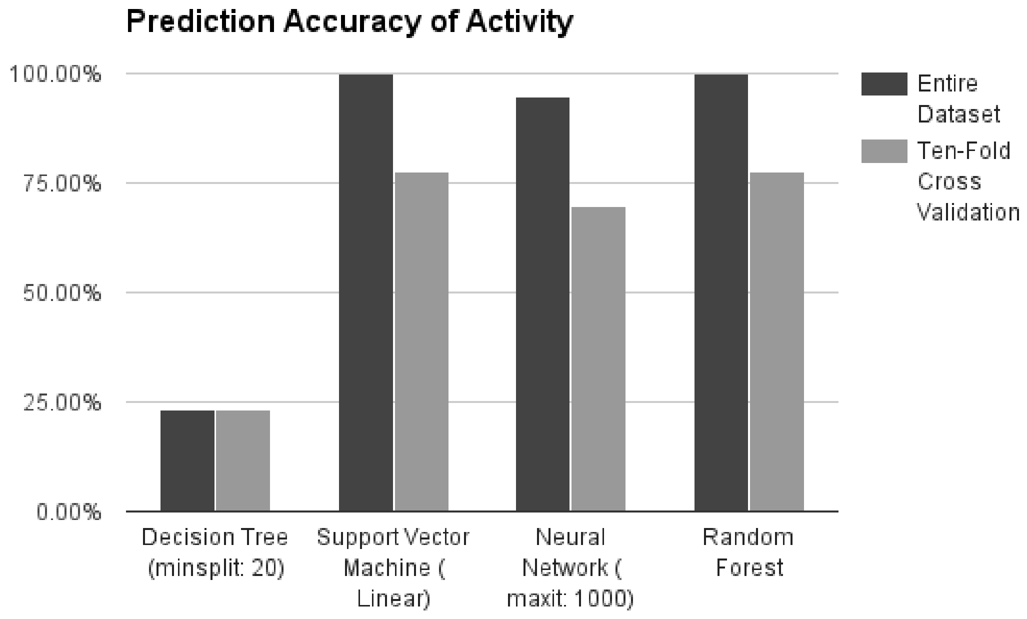

Figure 4.2: Comparison of prediction accuracy of activity for decision tree, SVM, neural network, random forest classifiers

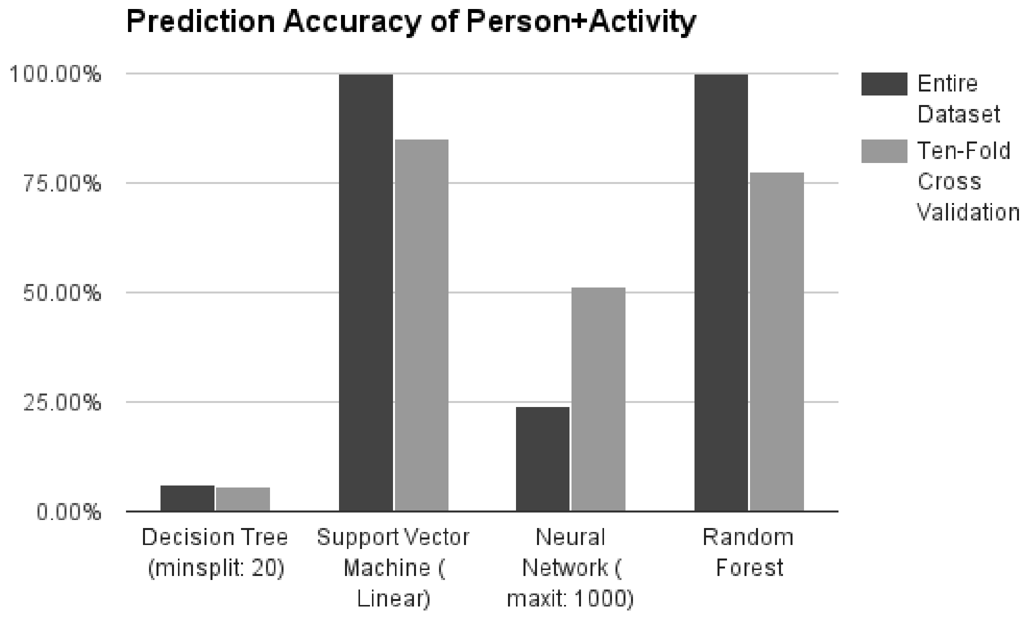

Figure 4.3: Comparison of prediction accuracy of person and activity for decision tree, SVM, neural network, random forest classifiers

Rank	Variable	Significance Score
1	B1.Beta.1	25.32
2	B3.Theta.2	8.29
3	B1.Delta.3	7.04
4	B8.Gamma.3	6.43
5	B4.Delta.1	4.04
6	B1.Gamma.1	3.93
7	B4.Alpha.1	3.74
8	B5.Delta.4	3.50
9	B2.Delta.1	3.07
10	B1.Beta.2	2.92

Table 4.5: Person - Most significant independent variables using random forest classifiers

Rank	Variable	Significance Score
1	B1.Theta.2	5.59
2	B6.Beta.3	4.66
3	B3.Delta.4	4.33
4	B1.Delta.3	4.10
5	B7.Delta.4	4.05
6	B8.Theta.1	3.49
7	B5.Beta.1	3.11
8	B8.Gamma.4	2.82
9	B8.Gamma.1	2.80
10	B2.Gamma.4	2.76

Table 4.6: Activity - Most significant independent variables using random forest classifiers

tree classifiers do not perform well using our knowledge representation of the EEG data and at best only achieve a precision of 65%.

Tables 4.5, 4.6, and 4.7 show the top ten significant variables obtained by the random forest classifier when classifying by Person, Activity, or Person and Activity, respectively. The variable names include the bin number, wave, and location. For example, bin 7 (85% to 95%) of the theta wave at location 4 would be named ‘B7.Theta.4’.

Rank	Variable	Significance Score
1	B7.Theta.4	6.43
2	B7.Theta.3	5.02
3	B6.Beta.2	4.33
4	B1.Delta.3	4.30
5	B4.Delta.1	4.26
6	B2.Delta.1	4.14
7	B3.Alpha.2	4.00
8	B7.Theta.1	3.75
9	B2.Alpha.1	3.72
10	B4.Alpha.1	3.51

Table 4.7: Person+Activity - Most significant independent variables using random forest classifiers

Rank	Variable	Significance Score
151	B1.Gamma.4	0.00
152	B2.Alpha.3	0.00
153	B3.Beta.4	0.00
154	B3.Gamma.4	0.00
155	B6.Alpha.4	0.00
156	B6.Delta.4	0.00
157	B6.Theta.4	0.00
158	B7.Beta.4	0.00
159	B7.Gamma.4	0.00
160	B8.Alpha.4	0.00

Table 4.8: Person - Least significant independent variables using random forest classifiers

Rank	Variable	Significance Score
151	B8.Alpha.4	0.01
152	B2.Delta.4	0.01
153	B1.Gamma.4	0.01
154	B8.Alpha.3	0.01
155	B3.Gamma.4	0.00
156	B1.Beta.4	0.00
157	B6.Alpha.4	0.00
158	B8.Delta.4	0.00
159	B6.Theta.4	0.00
160	B8.Theta.4	0.00

Table 4.9: Activity - Least significant independent variables using random forest classifiers

Rank	Variable	Significance Score
151	B6.Alpha.3	0.01
152	B3.Beta.4	0.01
153	B6.Theta.3	0.01
154	B6.Alpha.4	0.01
155	B6.Theta.4	0.00
156	B1.Beta.4	0.00
157	B1.Gamma.4	0.00
158	B8.Alpha.4	0.00
159	B6.Delta.4	0.00
160	B8.Theta.4	0.00

Table 4.10: Person+Activity - Least significant independent variables using random forest classifiers

The most significant independent variable for predicting a person, activity, or person and activity was bin 1 (0% to 5%) of beta wave at location 1 with a significance score of 25%, bin 1 (0% to 5%) of theta wave at location 2 with a significance score of 6%, and bin 7 (85% to 95%) of theta wave at location 4 with a significance score of 6%, respectively. The theta wave at location 2 is significant for predicting either person or activity with a significance score of 8% and 6%, respectively.

Tables 4.8, 4.9, and 4.10 show the bottom ten significant variables obtained by the random forest classifier when classifying by Person, Activity, or Person and Activity, respectively.

The least significant independent variable for predicting a person, activity, or person and activity was bin 8 (95% to 100%) of alpha wave at location 4 with a significance score of 0%, bin 8 (95% to 100%) of theta wave at location 4 with a significance score of 0%, and bin 8 (95% to 100%) of theta wave at location 4 with a significance score of 0%, respectively. The signals from locations 3 and 4 are shown to be often the least significant variable overall for predicting person, activity, or person

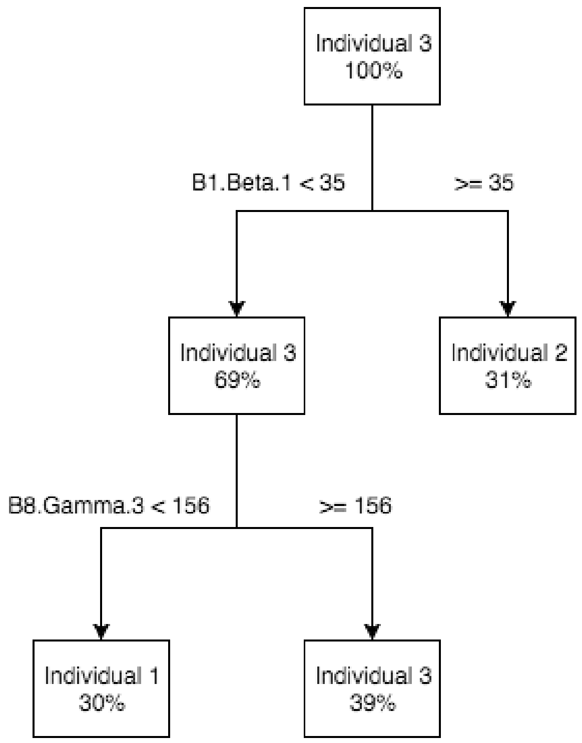

Figure 4.4: The decision tree for predicting a person

and activity with a significance score of approximately 0%. Locations 3 and 4 are the back right and back left sensors, respectively, where the occipital lobe is located. Locations 1 and 2 are the front right and front left sensors, respectively, where the frontal lobe is located. This could indicate that the back of the brain, the occipital lobe, is not as significant for differentiating the activities specified.

Figure 4.4 shows a graphical representation of the decision tree classifier that helps us understand the logical process of the person classification. Notice individual 0, with only 23 records instead of 50 records, is not included in the decision tree. This demonstrates that individuals with fewer records were neglected.

The most accuracy classifiers are SVM, neural networks, and random forests. Tables 4.11, 4.12, 4.13, 4.14, 4.15, and 4.16 show the confusion matrices for the SVM

	0	1	2	3
0	23	0	0	0
1	0	50	0	0
2	0	0	50	0
3	0	0	0	55

Table 4.11: Confusion Matrix - Person - SVM - Entire dataset

	0	1	2	3
0	11.13	0.00	0.56	0.56
1	0.00	27.55	0.00	0.59
2	1.14	0.59	27.58	0.59
3	0.59	0.00	0.00	29.13

Table 4.12: Confusion Matrix - Person - SVM - Ten-fold Cross Validation

classifier for classifying by Person or Activity or Person and Activity, respectively. Tables 4.17, 4.18, 4.19, 4.20, 4.21, and 4.22 show the confusion matrices for the neural network classifier for classifying by Person or Activity or Person and Activity, respectively. Tables 4.23, 4.24, and 4.25 show the confusion matrices for the random forest classifier for classifying by Person or Activity or Person and Activity, respectively. For the ten-fold cross-validation confusion matrices the entries are percentages of table totals.

Tables 4.12, 4.18, and 4.23 show correctly predicting individual 0 is not as likely as for the other persons, which can be expected since individual 0 has only 23 instead of 50 records.

	game	music	none	reading	video
game	41	0	0	0	0
music	0	35	0	0	0
none	0	0	34	0	0
reading	0	0	0	34	0
video	0	0	0	0	34

Table 4.13: Confusion Matrix - Activity - SVM - Entire dataset

	game	music	none	reading	video
game	21.40	1.15	0.00	2.89	3.41
music	0.00	13.50	1.67	0.00	0.00
none	0.00	2.75	15.70	0.53	0.00
reading	0.59	0.00	0.59	13.45	2.26
video	1.14	2.20	1.11	2.20	13.47

Table 4.14: Confusion Matrix - Activity - SVM - Ten-fold Cross Validation

		0					1					2					3				
		G	M	N	R	V	G	M	N	R	V	G	M	N	R	V	G	M	N	R	V
0	game	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	music	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	none	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	reading	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	video	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	game	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	music	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0
1	none	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0
1	reading	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0
1	video	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0
2	game	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0
2	music	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0
2	none	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0
2	reading	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0
2	video	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0
3	game	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0
3	music	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0
3	none	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0
3	reading	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0
3	video	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11

Table 4.15: Confusion Matrix - Person+Activity - SVM - Entire dataset

		0					1					2					3				
		G	M	N	R	V	G	M	N	R	V	G	M	N	R	V	G	M	N	R	V
0	game	5.63	0.00	0.00	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0	music	0.00	1.73	0.56	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0	none	0.00	0.00	0.56	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.62	0.00	0.00	0.00
0	reading	0.00	0.00	0.00	0.56	0.00	0.00	0.56	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0	video	0.00	0.00	0.00	0.00	1.11	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	game	0.00	0.00	0.00	0.00	0.00	4.58	0.56	0.56	0.00	0.56	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	music	0.00	0.00	0.00	0.00	0.00	0.00	3.44	1.18	0.00	1.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	none	0.00	0.00	0.00	0.00	0.00	0.00	0.50	3.90	0.00	0.56	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	reading	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.63	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	video	0.00	0.00	0.00	0.00	0.00	1.06	0.59	0.00	0.00	3.47	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	game	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.63	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	music	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.08	0.62	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	none	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.56	5.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	reading	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.08	0.00	0.00	0.00	0.00	0.00	0.00
2	video	0.00	0.56	0.56	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.56	5.63	0.00	0.00	0.00	0.00	0.00
3	game	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.55	0.00	0.00	0.59	0.00
3	music	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.01	0.00	0.00	0.00
3	none	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	6.13	0.00	0.00
3	reading	0.00	0.00	0.00	0.56	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.59	0.00	0.00	5.60	0.56
3	video	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.56	0.00	0.00	5.63

Table 4.16: Confusion Matrix - Person+Activity - SVM - Ten-fold Cross Validation

	0	1	2	3
0	23	0	0	0
1	0	50	0	0
2	0	0	50	0
3	0	0	0	55

Table 4.17: Confusion Matrix - Person - Neural Network (1000 iterations) - Entire dataset

	0	1	2	3
0	10.08	0.00	0.53	0.00
1	0.00	27.61	1.11	0.00
2	1.64	0.53	25.91	0.00
3	1.14	0.00	0.59	30.86

Table 4.18: Confusion Matrix - Person - Neural Network (1000 iterations) - Ten-fold Cross Validation

	game	music	none	reading	video
game	41	1	0	2	5
music	0	33	0	0	0
none	0	1	34	0	0
reading	0	0	0	32	0
video	0	0	0	0	29

Table 4.19: Confusion Matrix - Activity - Neural Network (1000 iterations) - Entire dataset

	game	music	none	reading	video
game	19.18	0.56	1.18	1.18	4.52
music	0.53	14.07	3.90	1.05	2.16
none	0.56	3.31	12.92	1.08	2.26
reading	0.00	0.53	0.56	14.69	1.21
video	2.85	1.15	0.56	1.08	8.92

Table 4.20: Confusion Matrix - Activity - Neural Network (1000 iterations) - Ten-fold Cross Validation

		0					1					2					3				
		G	M	N	R	V	G	M	N	R	V	G	M	N	R	V	G	M	N	R	V
0	game	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	music	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	none	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	reading	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	video	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	game	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	music	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	none	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	reading	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	video	0	0	0	0	0	0	10	0	10	10	0	0	0	0	0	0	0	0	0	0
2	game	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	music	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	none	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	reading	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	video	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	game	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	9
3	music	10	4	2	0	0	10	0	10	0	0	10	10	10	10	10	0	11	0	11	2
3	none	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0
3	reading	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	video	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4.21: Confusion Matrix - Person+Activity - Neural Network (1000 iterations) - Entire dataset

		0					1					2					3				
		G	M	N	R	V	G	M	N	R	V	G	M	N	R	V	G	M	N	R	V
0	game	3.84	0.00	0.00	0.00	0.53	0.53	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0	music	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.53	0.00	0.00	0.00	0.00	0.00
0	none	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0	reading	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0	video	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	game	0.00	0.00	0.00	0.00	0.00	1.77	1.11	0.00	0.00	0.00	0.59	0.00	0.00	0.00	0.00	0.00	0.56	0.00	0.00	0.00
1	music	0.00	0.00	0.00	0.00	0.00	0.59	2.39	1.70	0.00	2.39	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	none	0.00	0.00	0.00	0.00	0.00	0.00	0.53	2.78	0.00	1.11	0.00	0.00	0.00	0.00	0.00	0.00	0.59	0.00	0.00	0.00
1	reading	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.64	0.00	0.00	0.59	0.00	0.53	0.00	0.00	0.00	0.00	0.00	0.00
1	video	0.00	0.00	0.00	0.00	0.00	1.58	1.08	0.00	0.00	1.05	0.00	0.53	0.53	0.00	0.00	0.00	0.00	0.00	0.00	0.59
2	game	0.00	0.00	0.00	0.00	0.59	0.00	0.00	0.00	0.00	0.00	4.46	0.00	0.00	0.59	1.08	0.00	0.00	0.00	0.00	0.00
2	music	0.59	0.00	0.00	0.00	0.00	0.59	0.00	0.62	0.00	0.00	0.00	3.41	0.59	0.59	0.00	0.00	0.53	0.00	0.00	0.00
2	none	0.00	0.00	0.00	0.53	0.00	0.00	0.00	0.00	0.00	0.56	0.00	1.11	3.94	0.53	0.59	0.00	0.00	0.00	0.00	0.00
2	reading	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.53	0.00	0.00	0.59	0.00	0.00	3.41	0.00	0.00	0.00	0.00	0.00	0.00
2	video	0.62	1.14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.80	0.00	0.00	0.00	0.00	0.00
3	game	0.00	0.53	0.00	0.53	0.53	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.49	0.00	0.59	2.82	0.00
3	music	0.59	0.53	1.11	0.00	0.00	0.00	0.53	0.00	0.00	0.00	0.00	0.00	0.59	0.00	0.00	0.00	2.27	0.59	0.53	0.53
3	none	0.00	0.00	0.00	0.53	0.00	0.00	0.00	0.00	0.00	0.53	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.46	0.00	0.00
3	reading	0.00	0.00	0.56	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.11	1.67	1.11	0.53	1.68	1.11
3	video	0.00	0.00	0.00	0.00	0.00	0.59	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.53	0.00	1.18	0.00	1.14	3.94

Table 4.22: Confusion Matrix - Person+Activity - Neural Network (1000 iterations) - Ten-fold Cross Validation

	0	1	2	3
0	11.20	0.00	0.59	0.53
1	0.00	25.96	0.00	0.59
2	0.00	1.05	25.96	0.00
3	1.64	1.14	1.61	29.73

Table 4.23: Confusion Matrix - Person - Random Forest - Ten-fold Cross Validation

	game	music	none	reading	video
game	19.12	1.08	0.53	2.78	3.93
music	0.59	16.34	2.88	0.00	0.59
none	0.00	0.56	14.60	0.00	0.53
reading	0.56	0.56	0.00	14.02	2.32
video	2.82	1.14	1.11	2.25	11.70

Table 4.24: Confusion Matrix - Activity - Random Forest - Ten-fold Cross Validation

		0					1					2					3				
		G	M	N	R	V	G	M	N	R	V	G	M	N	R	V	G	M	N	R	V
0	game	5.64	0.00	0.00	1.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.53	0.00	0.00	0.00	0.00	0.00
0	music	0.00	0.00	0.59	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0	none	0.00	0.59	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0	reading	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0	video	0.00	0.00	0.00	0.00	1.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	game	0.00	0.00	0.00	0.00	0.00	4.58	1.09	0.50	0.00	1.11	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	music	0.00	0.00	0.00	0.00	0.00	0.00	2.33	1.14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	none	0.00	0.00	0.00	0.00	0.00	0.00	0.56	3.41	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.56	0.00	0.00
1	reading	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.64	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	video	0.00	0.00	0.00	0.00	0.00	1.06	1.67	0.00	0.00	3.94	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	game	0.00	0.00	0.00	0.00	0.56	0.00	0.00	0.00	0.00	0.00	5.14	0.00	0.00	0.56	0.00	0.00	0.00	0.00	0.00	0.00
2	music	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.64	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	none	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	reading	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.59	0.00	0.00	0.50	0.00	0.00	5.08	0.00	0.00	0.00	0.00	0.00	0.00
2	video	0.00	1.06	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.62	0.00	5.11	0.00	0.00	0.00	0.00	0.00
3	game	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.98	0.00	0.00	2.14	0.00
3	music	0.00	0.00	0.56	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.58	0.00	0.00	0.00
3	none	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.59	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.64	0.00	0.00
3	reading	0.00	0.59	0.00	0.56	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.21	0.00	0.00	2.78	0.00
3	video	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.59	0.00	1.21	6.19

Table 4.25: Confusion Matrix - Person+Activity - Random Forest - Ten-fold Cross Validation

Chapter 5

Semi-Supervised Evolutionary Learning

Table 5.1 shows the confusion matrix resulting from K-means clustering. The rows represent clusters and columns represent classes. The correspondence between clusters and classes was based on the most dominant class in a given cluster. It is clear that cluster 1 matches class 1 reasonably well. Most members of cluster 2 are from class 2. Cluster 0 is a little difficult to match with a class. Most of the members from class 0 belong to cluster 0. However, the most dominant class in cluster 0 is class 2. Since we had already assigned another cluster to class 2, we associated the cluster 0 with class 0. This assignment leads to the precision, recall, and, F-measure values shown in Table 5.2 and Figure 5.1.

Precision tells us the likelihood of us being correct when we predict a class. Precision can be calculated by dividing the diagonal value in a row by the sum of the row. With K-means clustering, our precision is quite high for persons 1 and 2. However, the value is very low for person 0. The likelihood of us classifying a given person in the correct cluster is given by recall. Recall is calculated by the diagonal value for a column and the sum of the column. Recall of K-means for persons 0 and 1 are fairly high, but quite low for person 2. F-measure is the harmonic mean of precision and

Cluster/Class	0	1	2
0	20	2	29
1	0	48	0
2	3	0	21

Table 5.1: Confusion Matrix - K-Means

Person	Precision	Recall	F-Measure
0	20/51 = 39%	20/23 = 87%	54%
1	48/48 =100%	48/50 = 96%	98%
2	21/24 = 88%	21/50 = 42%	57%

Table 5.2: Precision and Recall and F-Measure - K-means

recall. F-measure is calculated as:

$$F = 2 \times \frac{precision \times recall}{precision + recall} \tag{5.1}$$

Table 5.3 shows the confusion matrix resulting from the evolutionary semi-supervised K-medoids clustering. As before, we matched the clusters and classes based on the most dominant class in a given cluster. This assignment leads to the precision, recall, and F-measure values shown in Table 5.4 and Figure 5.2. In comparison to the precision, recall, and F-measure values from K-means shown in Table 5.2 and Figure 5.1, the evolutionary semi-supervised K-medoids provide more reasonable values. All the

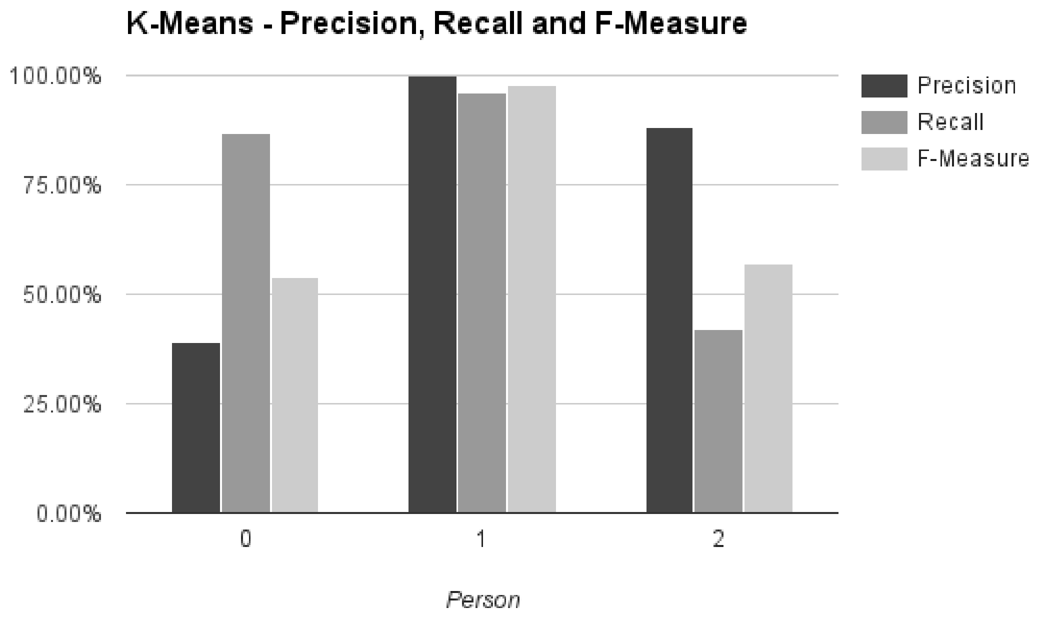

Figure 5.1: Precision and Recall and F-Measure - K-means

Cluster/Class	0	1	2
0	15	0	6
1	0	49	12
2	8	1	32

Table 5.3: Confusion Matrix
Semi-supervised K-Medoids

Person	Precision	Recall	F-Measure
0	15/21 = 71%	15/23 = 65%	68%
1	49/61 = 80%	49/50 = 98%	88%
2	32/41 = 78%	32/50 = 64%	70%

Table 5.4: Precision and Recall and F-Measure
Semi-supervised K-Medoids

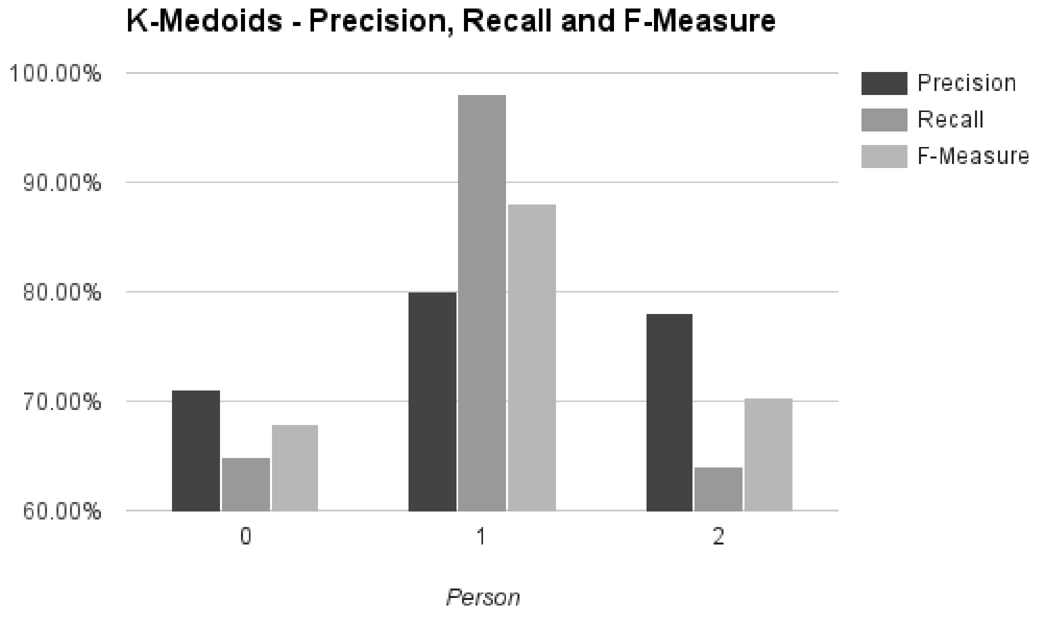

Figure 5.2: Precision and Recall and F-Measure - K-Medoids

Cluster/Class	0	1	2
0	18	0	6
1	3	50	13
2	7	13	49

Table 5.5: Confusion Matrix - Upper bound
Semi-supervised Rough K-Medoids

Cluster/Class	0	1	2
0	14	0	1
1	3	35	0
2	3	0	31

Table 5.6: Confusion Matrix - Lower bound
Semi-supervised Rough K-Medoids

precision, recall and F-measure values are above 64%, as opposed to the 39% and 42% values seen for precision of person 0 and recall of person 2 with K-means clustering.

Since rough clustering provides upper and lower bounds of clustering, we analyze them separately. It should be noted that the lower bounds are exclusive. A pattern belongs to a lower bound when we are almost certain of its membership. The upper bound, on the other hand, is inclusive. If there is a reasonable chance that a pattern may belong to a class, we assign it to the class's upper bound.

Table 5.5 shows the confusion matrix of upper bounds resulting from the evolutionary semi-supervised Rough K-medoids clustering. As before, we matched the

Person	Precision	
	Upper Bound	Lower Bounds
0	18/24 = 75.00%	14/15 = 93.33%
1	50/66 = 75.75%	35/38 = 92.10%
2	49/69 = 71.01%	31/34 = 91.17%

Table 5.7: Precision of clustering
Semi-supervised Rough K-Medoids

Person	Recall	
	Upper Bound	Lower Bounds
0	18/23 = 78%	14/23 = 61%
1	50/50 = 100%	35/50 = 70%
2	49/50 = 98%	31/50 = 62%

Table 5.8: Recall of clustering
Semi-supervised Rough K-Medoids

Person	F-Measure	
	Upper Bound	Lower Bounds
0	76%	74%
1	86%	80%
2	82%	74%

Table 5.9: F-Measure of clustering
Semi-supervised Rough K-Medoids

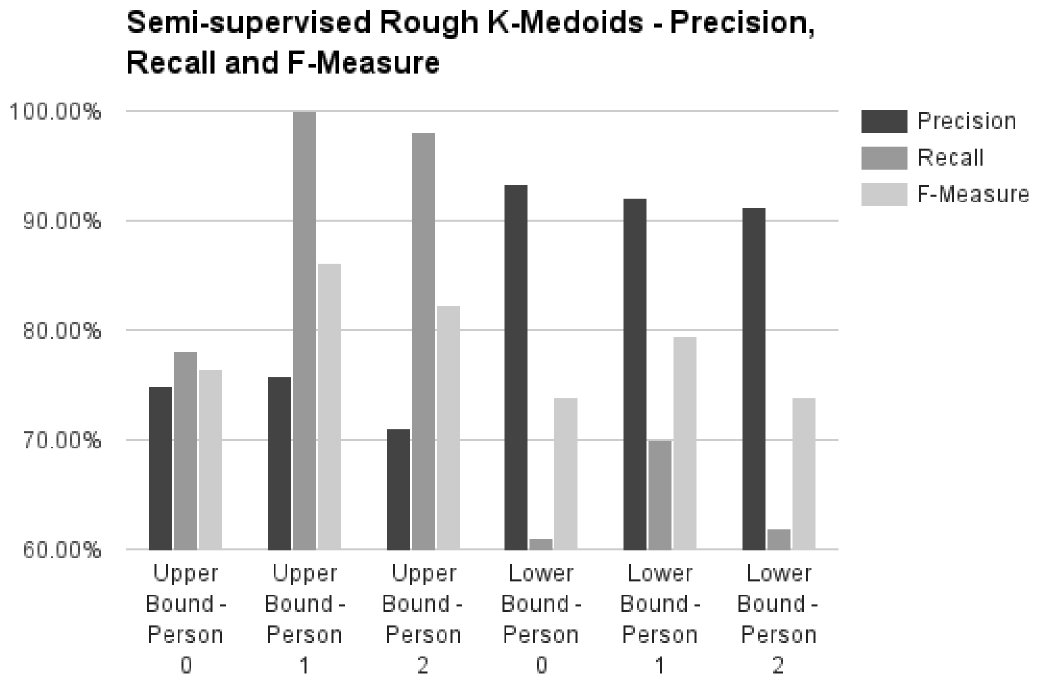

Figure 5.3: Precision and Recall and F-Measure - Semi-supervised Rough K-Medoids

clusters and classes based on the most dominant class in a given cluster. The cluster and class matching from the lower bounds was used to create the confusion matrix shown in Table 5.6. The precision values for the lower and upper bounds are shown in Table 5.7 and Figure 5.3. Since the boundary region represents ambivalence and has a higher likelihood of containing the wrong classes, the precision of the upper bounds is slightly lower than the corresponding K-medoids clustering. On the other hand, the precision of the lower bounds is significantly higher (91-93%) than any of the earlier clustering methods. The higher precision of the lower bound is due to its exclusive nature, while the inclusive nature of the upper bound leads to slightly lower precision.

The calculations of recall for the rough clustering shown in Table 5.8 and Figure 5.3 cannot use the sum of the columns because of the overlap between the upper bound clusters. Instead we divide by the cardinality of the known classification. The inclusive nature of upper bound provides us with higher recall values. On the other hand, the exclusivity of lower bounds leads to somewhat lower recall values.

Chapter 6

Summary and Conclusions

The advent of wearable technology is producing a large number of signals of varying length. These signals present exciting opportunities for data mining. This study demonstrates a supervised data mining process for data collected from various individuals performing different tasks. The data consists of electroencephalogram (EEG) brain signals from different points around the head collected using a Muse headband.

It was not possible to exactly control the duration of the activities even in an experimental setting. In the real-world, the signals will have even more variation in length. Most data mining techniques are based on fixed-length object representation. This paper shows that histograms of brain signals can be a very useful representation for data mining activities. One of the primary advantages of the histograms is that they reduce the variable length of signals to fixed length representations. That means a ten minute long brain signal will have the same representation as a one minute signal.

These representations were successfully used to identity persons with the following classifiers: support vector machines, neural networks, random forests, and semi-supervised crisp and rough K-medoids. The ten-fold cross-validation results varied from 95% to 93% for SVM, neural networks, and random forests, with decision trees providing the least precision with 65%. The results for predicting activities were a little less precise. However, SVM's precision of 78% for activities and 85% for person and activity showed that the proposed representation is quite credible for predicting activities from the brain signals.

The crisp semi-supervised K-medoids clustering had reasonable precision and recall values for predicting persons. Rough clustering provides upper and lower bounds of clustering, where the upper bounds are inclusive in nature. If a pattern has reasonable chance of belonging to a cluster, it goes into its upper bound. This leads to higher recall values. The lower bounds, on the other hand, are exclusive. A pattern goes into the lower bound of a cluster only if there is a very high chance that the pattern belongs to the cluster. This leads to lower recall and higher precision values for the lower bounds of clustering. An analyst can use the upper bounds of a cluster when higher recall is desired. Lower bounds of the clusters will be useful when the precision is an overwhelming criterion.

Bibliography

[1] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[2] G. T. Carolin Strobl, James Malley, "An introduction to recursive partitioning: Rationale, application, and characteristics of classification and regression trees, bagging, and random forests," *Psychological Methods*, vol. 14, no. 4, pp. 323–348, 2009.

[3] P. L. Zhibin Li *et al.*, "Using support vector machine models for crash injury severity analysis," *Accident Analysis and Prevention*, vol. 45, pp. 478–486, 2012.

[4] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[5] C. M. Bishop, "Neural networks and their applications," *Review of Scientific Instruments*, vol. 65, no. 6, June 1994.

[6] G. P. Pawan Lingras, "Rough clustering," *WIREs Data Mining and Knowledge Discovery*, vol. 1, January/February 2011.

[7] GAlib. Galib: A c++ library of genetic algorithm components. [Online]. Available: http://lancet.mit.edu/ga/dist/galibdoc.pdf

[8] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural computation*, vol. 10, no. 7, pp. 1895–1923, 1998.

[9] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[10] V. Svetnik, A. Liaw, C. Tong, J. C. Culberson, R. P. Sheridan, and B. P. Feuston, "Random forest: a classification and regression tool for compound classification and qsar modeling," *Journal of chemical information and computer sciences*, vol. 43, no. 6, pp. 1947–1958, 2003.

[11] G. Peters, "Evolutionary rough k-medoid clustering," *Transactions on Rough Sets*, vol. VIII, pp. 289–306, 2008.

[12] P. Lingras, "Evolutionary rough k-means algorithm," *Proceedings of Rough Set and Knowledge Technologies 2009, Lecture Notes in Computer Science 5589, Springer Verlag*, pp. 68–75, 2009.

[13] ——, "Rough k-medoid clustering using gas'," *Proceedings of ICCI 2009, Hong Kong*, 2009.

[14] D. Evans, "The internet of things. how the next evolution of the internet is changing everything." [Online]. Available: http://www.iotsworldcongress.com/documents/4643185/0/IoT_IBSG_0411FINAL+Cisco.pdf

[15] Pebble. Pebble sensors. [Online]. Available: https://developer.getpebble.com/guides/pebble-apps/sensors/

[16] Apple, "Apple watch." [Online]. Available: http://www.apple.com/ca/watch/

[17] Nymi. Nymi. [Online]. Available: https://www.nymi.com/

[18] Fitbit. Why fitbit. [Online]. Available: https://www.fitbit.com/ca/whyfitbit

[19] Jawbone. Jawbone trackers. [Online]. Available: https://jawbone.com/up/trackers

[20] Myo. Myo. [Online]. Available: https://www.myo.com/techspecs

[21] T. Labs. Myo tech specs. [Online]. Available: https://www.myo.com/techspecs

[22] A. Toor, "New google glass ui video shows off search, camera, and voice translation features." [Online]. Available: http://www.theverge.com/2013/2/20/4008180/google-glass-ui-previewed-in-new-video/in/2689413

[23] Microsoft. Microsoft hololens hardware. [Online]. Available: https://www.microsoft.com/microsoft-hololens/en-us/hardware

[24] D. Bohn and T. Warren. (2015, January) Up close with the hololens, microsoft's most intriguing product in years. [Online]. Available: http://www.theverge.com/2015/1/21/7868251/microsoft-hololens-hologram-hands-on-experience

[25] Muse, "What does muse measure?" [Online]. Available: http://www.choosemuse.com/what-does-muse-measure/

[26] Emotiv, "Emotiv specifications." [Online]. Available: https://emotiv.com/product-specs/Emotiv%20EPOC%20Specifications%202014.pdf

[27] ——, "Compare emotiv products." [Online]. Available: https://emotiv.com/store/compare/

[28] R. Gaddis, "What is the future of fabric? these smart textiles will blow your mind." [Online]. Available: http://www.forbes.com/sites/forbesstylefile/2014/05/07/what-is-the-future-of-fabric-these-smart-textiles-will-blow-your-mind/

[29] (2015) How it works - omsignal. [Online]. Available: http://www.omsignal.com/pages/how-it-works

[30] (2015) Hexoskin. [Online]. Available: http://www.hexoskin.com/

[31] (2015) They're using hexoskin. [Online]. Available: http://www.hexoskin.com/pages/theyre-using-hexoskin

[32] "Smart skin technologies." [Online]. Available: http://smartskintech.com/

[33] "E-textiles." [Online]. Available: https://en.wikipedia.org/wiki/E-textiles

[34] A. J. Myles *et al.*, "An introduction to decision tree modeling," *Journal of Chemometrics*, 2004.

[35] E. Fernández-Blanco *et al.*, "Random forest classification based on star graph topological indices for antioxidant proteins," *Journal of Theoretical Biology*, October 2012.

[36] Z.-X. Yang *et al.*, "Multiple birth support vector machine for multi-class classification," *Neural Comput & Applic*, vol. 22, no. 1, pp. S153–S161, 2013.

[37] M. Sun, "A multi-class support vector machine: Theory and model," *International Journal of Information Technology & Decision Making*, vol. 12, no. 6, pp. 1175–1199, 2013.

[38] R. M. Laura Auria, "Support vector machines (svm) as a technique for solvency analysis," *IDEAS Working Paper Series from RePEc*, 2008.

[39] (2016, April) ksvm kernlab - support vector machines. [Online]. Available: http://www.inside-r.org/node/63499

[40] M. Gevrey *et al.*, "Review and comparison of methods to study the contribution of variables in artificial neural network models," *Ecological Modelling*, vol. 160, pp. 249–264, 2003.

[41] R. A. McIndoe *et al.*, "Parakmeans: Implementation of a parallelized k-means algorithm suitable for general laboratory use," *BMC Bioinformatics*, April 2008.

[42] B. Aubaidan *et al.*, "Comparative study of k-means and k-means++ clustering algorithms on crime domain," *Journal of Computer Science*, vol. 10, no. 7, pp. 1197–1206, 2014.

[43] Z. Pawlak, "Rough sets: Theoretical aspects of reasoning about data," *Kluwer Academic Publishers*, 1992.

[44] ——, "Rough sets," *International Journal of Information and Computer Sciences*, vol. 11, pp. 145–172, 1982.

[45] Y. Yao, "Constructive and algebraic methods of the theory of rough sets," *Information Sciences*, vol. 109, pp. 21–47, 1998.

[46] A. S. L. Polkowski, "Rough mereology: A new paradigm for approximate reasoning," *International Journal of Approximate Reasoning*, vol. 15, no. 4, pp. 333–365, 1996.

[47] J. S. A. Skowron, "Information granules in distributed environment," *Zhong N, Skowron A, Ohsuga S (eds), New Directions in Rough Sets, Data Mining, and Granular-Soft Computing, Lecture notes in Artificial Intelligence*, vol. 1711, pp. 357–365, 1999.

[48] Y. Yao, "Interval-set algebra for qualitative knowledge representation, proceedings of the 5th international conference on computing and information," *Chang, C.K., and Koczkodaj, W.W. (eds.), IEEE Computer Society Press*, pp. 370–375, 1993.

[49] J. Holland, "Adaptation in natural and artificial systems," *University of Michigan Press, Ann Arbor*, 1975.

[50] F. P. B.P. Buckles, "Genetic algorithms," *IEEE Computer Press, Los Alamitos, California*, 1994.

[51] Interaxon. (2016) Protocols: Compressed eeg packets. [Online]. Available: http://developer.choosemuse.com/protocols/bluetooth-packet-structure/compressed-eeg-packets

[52] ——. (2016) Research tools: Available data, relative power bands. [Online]. Available: http://developer.choosemuse.com/research-tools/available-data#Relative_Band_Powers

[53] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14, no. 2, 1995, pp. 1137–1145.

[54] S. N. Sivanandam and S. N. Deepa. (2007) Introduction to genetic algorithms. [Online]. Available: https://xa.yimg.com/kq/groups/86541084/1097235234/name/Introduction+to+Genetic+Algorithms,+Springer+(2008),+354073189X.pdf

[55] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd international conference on Machine learning.* ACM, 2006, pp. 233–240.

[56] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," 2011.

[57] J. T. Townsend, "Theoretical analysis of an alphabetic confusion matrix," *Perception & Psychophysics*, vol. 9, no. 1, pp. 40–50, 1971.