University of Tartu
Faculty of Mathematics and Computer Science
Institute of Computer Science
Information Technology

Stepan Bolotnikov

Usage of Fuzzy Classification Algorithms in Brain-Computer Interfaces

Bachelor's thesis (6 ECTP)

Supervisor: Ilya Kuzovkin, MSc
Seminar supervisor: Margus Niitsoo, PhD

Tartu 2014

Hägusate klassifikatsioonialgoritmide kasutamine aju-arvuti liidestes

Lühikokkuvõte:

Selles lõputöös uuritakse hägusate klassifikatsioonialgoritmide kasutamist elektroentsefalograafial (*electroencephalography*, EEG) põhinevates aju-arvuti liidestes (*brain-computer interfaces*, BCI). Uuritakse olemasolevat kirjandust BCI-des kasutatavate klassifikatsioonialgoritmide, hägusate algoritmide olemuse ja nende kasutamise kohta BCI-des. Hägusate algoritmide potentsiaalsete eeliste demonstreerimiseks realiseeritakse lihtne aju-arvuti liides, mis võimaldab kasutajal liigutada kursorit arvuti ekraanil. Testid selle rakendusega näitavad, et hägusad algoritmid sellist tüüpi rakendustes ei oma eelist traditsiooniliste algoritmide üle.

Võtmesõnad:

Aju-arvuti liidesed, klassifikatsioonialgoritmid, hägus klassifitseerimine

Usage of fuzzy classification algorithms in brain-computer interfaces

Abstract:

In this thesis, the usage of fuzzy classification algorithms in brain-computer interfaces (BCI) based on electroencephalography (EEG) is researched. We review the existing literature on BCI, the traditional crisp algorithms often used in BCI for classification, fuzzy classification algorithms and their application in BCI. A simple BCI system is implemented that allows the user to move a cursor on the computer screen. Tests conducted with this application show that fuzzy classification algorithms do not have advantage over crisp classification algorithms in this kind of BCI systems.

Keywords:

Brain-computer interfaces, classification algorithms, fuzzy classification

Contents

Introduction

Brain-computer interfaces (BCI) allow for a direct communication pathway between a human user's brain and an external device, enabling the user to operate the device without traditional interfaces like keyboards or mice. Research in this field began in the 1970s at the University of California, Los Angeles. Modern brain-computer interface research is primarily focused on neuroprosthetics, aiming to create technologies for restoring hearing, sight and mobility in physically disabled humans.

Brain-computer interfaces vary in invasivity from implants placed in or onto the human brain to completely non-invasive methods like electroencephalography (EEG) that use a set of electrodes fastened to the scalp to record electrical activity of the brain and attempt to interpret the recorded brainwaves based on previous knowledge. EEG is the most studied non-invasive technique partially because of its mobility, ease of use and low cost.

Traditionally, the classification algorithms in EEG-based BCI are only used to interpret each instance of data as one of the predefined classes or commands. Classification algorithms that utilize fuzzy logic instead of traditional crisp logic could potentially give more insight into the commands given by the user and even process multiple commands at the same time, for example moving the cursor on screen in a diagonal direction instead of only allowing a series of up-down and left-right motions.

In this thesis, the author researches the existing material on EEG and fuzzy classification algorithms and tries to determine whether the implementation of fuzzy logic in place of traditional methods in BCI is feasible and what benefits it provides.

The technological background is given in Chapter 1, introducing brain-computer interfaces based on electroencephalography and traditional classification algorithms used in this field.

Chapter 2 introduces fuzzy classification algorithms. In the beginning, we determine what are the main differences between crisp and fuzzy logic. After that we review the history of previous usage of fuzzy classification algorithms in BCI and describe some algorithms in greater detail.

Lastly, in Chapter 3, we implement a simple EEG-based BCI application using a fuzzy classification algorithm for moving the cursor on a computer screen. We explain the choice of technology, implementation process and conduct a series of experiments to determine if a fuzzy classifier in this kind of BCI can be better suited than a crisp classifier.

Chapter 1

Technological background

1.1 Electroencephalography in brain-computer interfaces

Electroencephalography (EEG) is a technique of recording the electrical activity in the brain with the help of electrodes placed on the scalp. EEG research on humans began with the work of Hans Berger in 1924, who recorded the first human electroencephalogram and gave the device its name.

EEG has been primarily used for diagnosing epilepsy[AKM05] and various other focal brain disorders[eeg13]. Even though the use has declined due to emergence of technologies with a higher spatial resolution, like Magnetic Resonance Imaging (MRI) and Computed Tomography (CT), EEG continues to be a valuable technique, especially where the requirements for high temporal resolution outweigh the need for spatial precision.

In brain-computer interfaces (BCI), EEG is the most-studied non-invasive communication method due to multiple factors, including the relatively low cost, high mobility of the devices and a quick setup.

An electroencephalogram is produced by monitoring electrical signals that are generated by the changes of electrical charge in the membrane of neurons - electrically excitable cells that comprise the core of the nervous system. Neurons transmit information with electrical or chemical signaling via special junctions between neurons called synapses. An impulse passing through a synapse causes the voltage on the membrane of the neuron to change, which generates an electrical flow in the membrane.

Most of the current generated by neurons' activity remains in the brain, but a small fraction still gets through the scalp in different areas, allowing us to measure the activity of different parts of the brain and interpret this data.

BCI uses machine learning techniques to train the interface about the kinds of actions that should be associated with a certain brain activity. This allows the interface to recognise known patterns in new data and classify them as the appropriate activities. For example, a simple BCI for controlling a cursor on a computer screen can be trained for commands "up", "down", "left" and "right" by asking the test subject to give those commands to a computer while connected to an EEG device. When later the subject attempts to reproduce these commands, similar brain activity is read by the electrodes, the interface recognises the signal as one of the commands and performs the action associated with it.

1.2 Classification algorithms in brain-computer interfaces

While some research[MW05] suggests that using regression algorithms can provide better results in certain usage cases, most BCI use classification algorithms for transforming user input into discrete commands. Classification algorithm is provided with a dataset in which each instance is assigned a class and is concerned with assigning classes to new unclassified units of data[IHW11].

In the context of BCI, when choosing an algorithm, two main domain-specific problems arise:

- curse of dimensionality,
- bias-variance dilemma.

In machine learning, the curse of dimensionality refers to the fact that as the dimensionality of the data increases, the predictive power of a classifier decreases, needing more and more training instances to produce classification of sufficient quality. In BCI the training sets are indeed often relatively small and the feature vectors have high dimensionality, therefore the curse of dimensionality must be accounted for. It's usually suggested to use five to ten times as many training samples for each class as the dimensionality of the feature vectors, otherwise the classifier will probably give poor results[RJ91].

There are generally three sources of error in classification problems:

- The natural noise within the system being observed.
- Bias - the divergence of the estimated mapping of instances to classes and the best possible mapping.
- Variance - sensitivity to the training set.

In order to have the lowest possible error rate, both bias and variance must be low, but there is a natural tradeoff called the bias-variance dilemma. Stable classifiers usually have high bias and low variance and unstable classifiers have low bias but high variance. Stabilization techniques, such as regularization and combination of classifiers, can be used to reduce variance.

The combination of classifiers refers to using several different classifiers instead of just one and aggregating the results in one of three ways:

- Boosting - each classifier focuses on the errors made by previous classifiers.
- Voting - each classifier determines a class and the class chosen by the majority is assigned.
- Stacking - using several so-called "level-0" classifiers and a meta-classifier that classifies the instance based on the results of all the level-0 classifiers.

Next we review some popular classification algorithms that have been successfully applied in BCI research.

1.2.1 Linear classifiers

Probably the most popular choice in BCI are linear classifiers[LCL+07] - algorithms using linear function for distinction between classes. Two main linear classifiers that have been used are linear discriminant analysis (LDA) and support vector machines (SVM) with linear kernel.

LDA uses a linear function to find hyperplanes separating instances into different classes. It is created by finding a projection that maximizes the distance between the means of different classes while minimizing the variance within classes.

With LDA, the one versus the rest (OVR), also known as one-against-all (OAA) strategy is often used in multi-class problems (as opposed to two-class problems), which consists of separating each class from all the others to find the hyperplane separating it from others. It has low computational requirements, is simple to use and generally provides good results. The main disadvantage of LDA is the implied linearity, which can potentially cause it to perform poorly on complex nonlinear data produced by EEG[GEV03].

A special version of LDA, called regularized Fisher's LDA (RFLDA) has also been used in BCI. It uses a regularization parameter that allows the classifier to better process outliers in the data and generally get better generalization capabilities. As outliers are common in data generated by EEG, this technique could give better results in BCI than the regular kind of LDA, but so far it has been far less popular.

SVM has a number of similarities to LDA. It also uses a hyperplane to separate instances into classes and like RFLDA it also has a regularization parameter, allowing errors on training set and accomodation to outliers. Like LDA, it's generally applied to multi-class problems with OVR strategy.

Unlike LDA, the hyperplane is found by maximizing the margins - distance between the hyperplane and the nearest training instances.

SVM has been known to perform well and have good generalization. It is possible to make it even more efficient with just a small increase of complexity by using the kernel trick - implicitly mapping the training set to another set using a kernel function. In BCI, a popular choice for a kernel function is the Gaussian or radial basis function (RBF).

SVM using this technique is called Gaussian SVM or RBF SVM.

SVM, although slower than LDA[LCL+07], has several important advantages - margin maximization and regulation term give SVM good generalization properties, decrease sensitivity to overtraining and the curse of dimensionality[BC00].

1.2.2 Neural networks

Neural networks (NN) are another popular category of classifiers in BCI. NN is presented as a system of interconnected artificial neurons - nodes that can receive, compute and output data and feed information through the network. This structure enables NN to produce nonlinear boundaries between the classes[Bis95]. As NN can be used to classify any number of classes, they are considered very flexible classifiers and have been successfully used in all kinds of BCI problems[LCL+07].

The most popular NN in BCI research is the Multilayer Perceptron (MLP)[LCL+07]. It is composed of several layers of neurons - input and output layers and one or more hidden layers. Every neuron is connected to the output of the neurons from the previous layer and the output layer determines the class of an instance. The one major drawback that is worth noting is that MLP is sensitive to overtraining, especially when noisy and non-stationary data is used as is the case with EEG.

Gaussian classifier is another NN worth noting as it was created specifically for BCI purposes. Each node of the Gaussian classifier is a Gaussian discriminant function that represents a class. The creators claim that the Gaussian classifier outperforms MLP on EEG data and can efficiently reject samples classification of which is not certain[dRMMF+02]. This classifier has also been successfully used in various BCI research cases.

Several other types of NN, for example Fuzzy ARTMAP NN, have been used in BCI, but with less significance[LCL+07].

1.2.3 Nonlinenar Bayesian classifiers

In BCI, two nonlinear Bayesian classifiers have been applied in various research - Bayes quadratic and the hidden Markov model (HMM). Another, Bayesian graphical network (BGN) has also been attempted, but is not as common as the others and at present is too slow for use in real-time BCI.

Bayesian classifiers produce nonlinear boundaries and are able to perform quite efficient rejection of uncertain samples, but they are still far less popular than linear classifiers and NN[LCL+07].

In these classifiers, the Bayes rule is used to calculate the probability of each instance to fall into each class and then the highest probability is used to assign the class[IHW11].

The Bayes quadratic assumes a different normal distribution of data. While not very popular in BCI research, it has been successfully applied to motor imagery and mental task classification.

HMM can be seen as a probabilistic automaton, representing a probability of a given sequence of feature vectors, each state of which can modelize the probability of a feature vector occuring. In BCI, Gaussian mixture models (GMM) are usually used as the probabilities. HMM are not widespread in BCI research, but it has been shown that they are a perfect fit for classifying time series, which can be beneficial, as temporal occurence is an important factor in EEG data.[Rab89].

1.2.4 Nearest neighbor classifiers

Nearest neighbor classifiers assign a class to a feature vector according to the nearest neighbors. In the case of k nearest neighbors (kNN), the neighbors are actual feature vectors from the training set whereas classifiers based on Mahalanobis distance assign a class that corresponds to the nearest class prototype, according to a metric called the Mahalanobis distance.[CST+03]

kNN attempts to assign a class to a new instance based on its k nearest neighbors from the training set. This technique isn't popular in the BCI research community, partially due to its sensitivity to the curse of dimensionality. It has, however, been successfully used in BCI systems with low-dimensional feature vectors.[BMBB04]

1.3 Fuzzy set theory

Fuzzy logic is a kind of many-valued logic, a logic in which there are more than two truth values. Unlike traditional binary logic, where a variable can take on one of two possible values (0 or 1, where 0 is false and 1 is true), in fuzzy logic

a variable may have any value that is between 0 and 1.[PM99] This allows for a representation of partial truth, which in many cases is more natural and closer to the way humans perceive the surrounding world and as such has applications in various fields like artificial intelligence.

Fuzzy logic deals with fuzzy sets, which extend the traditional understanding of a mathematical set.

A fuzzy set is a pair (U, m), where U is a set of objects and $m : U \to [0, 1]$ is a membership function that maps each object to a membership grade. It is said that for $x \in U$, if $m(x) = 1$ then x is fully included and if $m(x) = 0$, it is not included in the set U. For each $0 < m(x) < 1$, x is said to be a fuzzy member of the set U.

A simple way to understand fuzzy logic is to imagine the classification of temperatures or colours by humans - say we wish to divide the temperatures of water into three categories - "cold", "warm" and "hot". Probably everyone would agree that water with the temperature 90°C is "hot", so this temperature might be considered fully included in the set "hot", but for many other temperatures it's debatable whether they should be considered "hot" or "warm", so for example 40°C could be said to be a fuzzy member of the set "hot" with a membership degree of 0.5 and a fuzzy member of the set "warm" with a membership degree of 0.5.

This thesis deals with fuzzy classification algorithms that utilize fuzzy logic in their classification process, allowing for simultaneous assignment of several class labels with different membership grades to one data instance.

1.4 Fourier transform

The Fourier transform is a mathematical transformation used to transfer signals between time or spatial and frequency domains. It is very useful in all kinds of signal processing problems as it is fully reversible, allowing transformation from either of the domains into the other and therefore can greatly improve understanding of a signal.

The electric activity of the brain is composed of periodic signals that are sometimes called brain waves, so the data being recorded and interpreted in EEG is by nature periodic. This allows us to view the activity as a periodic signal and use the Fourier transform to find the discrete set of simple frequencies that compose this signal. Such technique allows for inspection of a series of successive EEG measurements as a single unit of data.

Chapter 2

Fuzzy classification algorithms

2.1 Principles and characteristics

Traditional classifiers, like the ones introduced in Section 1.2, typically use binary logic and assign so-called crisp labels. This means that classes are mutually exclusive and only one class can be assigned to each instance.

Fuzzy classifiers use fuzzy logic or fuzzy sets in their training and decision process. This allows for assignment of so-called soft labels. No longer is one class assigned fully to an instance, instead the instance can be a member of all classes simultaneously. This is done by assigning fuzzy pairs instead - pairs that contain the class being assigned and the probability that the instance is a member of said class.

Fuzzy classification represents knowledge more naturally to the way of human thinking and is more robust in tolerating imprecision, conflict, and missing information. [HL98]

2.2 Popular fuzzy classification algorithms

Many papers have been written on various attempts of "fuzzifying" traditional classification methods - modifying the algorithms to either work with fuzzy input or produce fuzzy decisions, as described previously. This section describes some approaches that we decided to view in detail.

2.2.1 Naïve Bayes

Naïve Bayes is a very simple probabilistic classifier. It uses the Bayes' theorem to calculate the probability of class membership for each instance while assuming that all the features of an instance are equally important in the decision process and that they are all independent. It calculates the probability for all classes and assigns the class that gets the highest probability. Both are very naïve assumptions, but the classifier has been known to provide suprisingly good results in various machine learning problems.

The naïve Bayes is designed to work with nominal features that have a finite number of possible values, not real numbers, but several techniques have been proposed for working with numeric data.

The simplest solution to the problem is to discretize the numeric features - divide them into a finite number of groups and work explicitly with those groups. A more advanced technique is to apply a clustering algorithm to the data, dividing it into clusters instead of equal groups and then using the transformed data to train the naïve Bayes.

For example, a study has suggested to use the fuzzy c-means clustering algorithm in conjuction with naïve Bayes in machine learning problems that deal with continuous features.[TPLX02]

Another solution is to calculate the probability of class membership with a suitable probability density function - a function describing the relative likelihood for the feature to have the set value - instead of Bayes theorem when dealing with numeric features[IHW11]. In many cases, using the probability density function for normal distribution is enough:

$$f(x) = \frac{1}{\sqrt{s\pi}\sigma} e^{\frac{(x-\mu)^2}{2\sigma^2}}$$

Usually, the naïve Bayes is used to assign just one class label to each feature vector - the one that gets has the highest probability. However, a simple way of constructing a fuzzy classifier would be to view all the probabilities calculated by the naïve Bayes as membership grades of the classes. This gives us additional insight into the nature of the data being classified, allowing us to draw further conclusions from it.

2.2.2 Fuzzy neural networks

As discussed previously, traditional neural networks work on binary logic, but several studies have been conducted on ways to include fuzzy set theory into artificial neural networks.

For example, Hon Keung Kwan and Yaling Cai[KC94] propose a four-layer fuzzy

NN and apply it to text recognition problems.

The NN was trained with 16×16 pixels images of the 26 letters of the English alphabet and 10 arabic numerals. The original patterns were then shifted in eight directions first by 1 pixel and then by 2 pixels. The NN, trained with the original 36 training patterns, was able to recognise all of the original patterns with 100% accuracy, patterns shifted by 1 pixel with 100% accuracy and patterns shifted by 2 pixels with an average accuracy of 92.01%. The NN was then trained with original patterns as well as 72 shifted patterns and was then able to recognise original patterns and those shifted by 1 pixel with 100% accuracy and patterns shifted by 2 pixels with 98.61% accuracy.

Their network works with discrete data and produces nonfuzzy (crisp) output, but uses fuzzy logic internally, fuzzifying the input pattern and calculating the similarities of the given input pattern to all the learned patterns before selecting the learned pattern with the highest similarity score and assigning it as the label of the input pattern.

It is possible to modify the algorithm to omit the last step that defuzzifies the data and outputs only the label that is determined to be the most accurate and inspect the similarity scores for all of the learned patterns. The authors also write that the algorithm can be modified for use in other pattern recognition problems.

The more popular fuzzy NN's are the fuzzy adaptive resonance theory (ART) and the fuzzy ARTMAP models.

Fuzzy ART extends the ART 1 neural network, based on adaptive resonance theory[Gro03] models by implementing fuzzy logic into the pattern recognition. Fuzzy ART also includes an optional but very useful feature - a normalization procedure called complement coding that allows for including the absence of features in feature vectors into the pattern classification process.[CGR91b]

It has been said that both ART 1 and fuzzy ART neural networks are not consistent, because the results critically depend on the order in which data is being processed. This effect can be somewhat reduced by using a slower learning rate, but cannot be completely removed.[Sar95]

ARTMAP, also known as predictive ART, is a hierarchical neural network architecture that is built from up from two slightly modified ART 1 or ART 2 modules. During training, the first ART module receives the feature vectors and the second receives their correct labels. Predictions are made in the first module and compared to the correct values given to the second module, allowing the network to rapidly self-organise.[CGR91a]

Fuzzy ART models have been incorporated into ARTMAP, creating the fuzzy ARTMAP architecture. Just like the original ARTMAP, fuzzy ARTMAP is a rapidly self-organising hierarchical architecture that works with fuzzy logic, like fuzzy ART.[CGM$^+$92]

In the original paper that introduced fuzzy ARTMAP, the NN was tested in four situations:

- Identifying which points of a square lie inside and which lie outside of a circle with an area half that of the square. After one training epoch with 100 exemplars, fuzzy ARTMAP achieved accuracy of 88.6% on the training set. At 100000 exemplars, the accuracy increased to 98.0%.

- Learning to tell two spirals apart. After one training epoch with 194 training points, fuzzy ARTMAP achieved a 100% accuracy. With 47 training points, the accuracy was 96.4%.

- Incremental approximation of the function $f(a) = (\sin 2\pi a)^2$ for $0 \leq a \leq 1$. After training the fuzzy ARTMAP with 50 exemplars, the accuracy of 72.3% was achieved.

- Optical text recognition. The best achieved simulation had a 94.7% correct prediction rate on a 4000-item dataset, showing lower error rates than in the original study[FS91] that proposed this simulation as a benchmark.

2.2.3 Fuzzy support vector machines

With SVM, multi-class problems are often solved by applying the OVR strategy. For each class, a hyperplane is found that best separates the instances of that class from all the others. When classifying a new feature vector, its position to all the separating hyperplanes is found and ideally the feature vector will fall on the "correct" side of only one of the hyperplanes and be assigned the label of that class. A problem arises when more than one classes claim the instance or when none of them do.

A paper by Takuya Inoue and Shigeo Abe[IA01] proposes fuzzy support vector machines (FSVM). Their technique defines a fuzzy membership function for each class based on the decision functions generated by the SVM. If a feature vector falls into only one class, the classification works just like with conventional SVMs. Otherwise, distances to all the separating hyperplanes are calculated. If the feature vector falls into more than one class, it is assigned to the class with the biggest distance and if it falls into none, it's assigned to the class with the smallest distance. It is possible to skip the last step and instead view all the calculated distances to gain more insight into the nature of the data that is being classified.

They evaluate the performance of the FSVM on three datasets. On thyroid dataset[WK90] FSVM performed better than conventional SVM with all the tested kernels. For example, with the dot kernel, conventional SVM achieved accuracy of 93.03% whereas FSVM achieved accuracy of 95.27%.

With blood cell data[Has88] and using the dot kernel the conventional SVM achieved accuracy of 67.58% and FSVM achieved accuracy of 85.38%.

With the hiragana dataset[TAT+91], the conventional SVM got to 82.86% accuracy and FSVM got to 93.32% accuracy.

In a latter study, the authors extend their method to another popular technique for handling multi-class problems with SVM - pairwise classification. In this technique, a separating hyperplane is learned for each pair of classes. The SVM was tested on the same datasets as in the previous study. On thyroid data, their pairwise FSVM achieved 96.62% accuracy with the polynomial kernel whereas a conventional pairwise SVM achieved 96.27% accuracy. On blood cell data the accuracy was 92.35% and 91.26% respectively. Hiragana dataset with 13 inputs yielded accuracy of 99.63% and 99.46% respectively and the hiragana dataset with 50 inputs yielded accuracy of 98.24% and 98.00% respectively.[AI02]

Another paper by Chun-Fu Lin and Sheng-De Wang[LW02] describes FSVM as an extension of SVM that deals with fuzzy data. Each feature vector is associated with a membership value, so that it's possible to for example mark noisy, uncertain instances as less important than other instances. Each feature vector then makes different contribution to the generation of the separating hyperplane. The usefulness of such approach is demonstrated in three experiments, including giving newer feature vectors more weight than to older ones, giving more weight to one of the classes and using class centers to reduce the weight of outliers. Authors claim that this technique allows for usage of SVM in more problem types and close with proposition that further research must be done to automatically or adaptively determine a fuzzy membership function for the input data.

This technique is further reviewed in a paper by Tai-Yue Wang and Huei-Min Chiang[WC07], where they introduce OAA-FSVM concept based on Lin and Wang's FSVM that is designed for multi-class problems. Their OAA-FSVM is tested in text categorization and it is shown that OAA-FSVM is more accurate than conventional OAA-SVM.

2.2.4 Fuzzy k-nearest neighbor

James Keller, Michael Gray and James Givens write in a paper[KGG85] that one of the problems of the traditional k nearest neighbor algorithm is that each training feature vector is assumed to be equally important in assigning a class label to a new feature vector, which can cause problems in some situations, especially where classes overlap, because outliers, atypical feature vectors, are given as much weight in decision as the vectors that are truly representative of the cluster.

They propose a fuzzy kNN algorithm for dealing with those situations and show that it outperforms the standard kNN algorithm. For example, on the iris dataset using 9 nearest neighbors for classification a crisp kNN algorithm misclassified 6

out of 150 feature vectors whereas the fuzzy kNN misclassified 4.

In their algorithm, the distance from the feature vector being classified to the neighbors that are being used in classification is used as a weight, giving nearer neighbors more importance in the decision process. They also propose several different ways of giving membership degrees to labeled feature vectors - crisply labeling them as full members of their assigned class and non-members of other classes, calculating membership degrees based on the distance from the nearest neighbors of the same class and calculating membership degrees based on distance from class mean.

2.3 Previous research of fuzzy classification in EEG-based BCI

The research of usage of fuzzy classification algorithms in BCI is limited, but certain papers definitely deserve to be mentioned.

In a paper[CMP08], Damien Coyle et al. have analysed the neural-time-series-prediction-preprocessing (NTSPP) framework, in which a self-organising fuzzy neural network (SOFNN) is used to preprocess raw EEG data. SOFNN is trained to specialize in predicting the EEG time-series, after which features are extracted from the signals predicted by the SOFNN. It has been shown that features extracted in this way are easier to classify than raw EEG data. The authors demonstrate that a SOFNN-based NTSPP framework is effective in multi-class BCI applications.

In another paper[CPM09] they propose a number of modifications to the design of the learning algorithm of the SOFNN used in NTSPP that are shown to greatly reduce computational costs. They demonstrate their work with a two-class EEG-based BCI. Furthermore, they perform a sensitivity analysis of the parameters of the SOFNN using EEG data recorded from three subjects in a left/right motor-imagery-based BCI experiment to create a general set of parameters that allows for creation of a completely parameterless BCI application that is capable of autonomous adaptation.

A paper[PPNS02] by Ramaswamy Palaniapan et al. proposes a new design for a EEG-based BCI built around a fuzzy ARTMAP neural network. The aim of their design is to classify the three best out of five available mental tasks for each test subject and they show that the design is accurate and can be successfully used in a tri-state switching device. To demonstrate this, they implement a tri-state Morse code scheme, that recognises three mental tasks - a dot, a dash and a space. In their implementation the construction of language is only dependent on the sequence of mental tasks, not their duration. They propose that this system could be further developed to provide communication for paralysed patients.

A paper[Lot06] by Fabien Lotte proposes a BCI design that uses a fuzzy inference system for classification of EEG data in a two-class BCI application based on motor imagery. The inference system reached accuracy (79%) comparable to a multilayer perceptron (78.9%) and SVM (79.4%) and outperformed a linear classifier (76.2%). The average computation time for classification of a feature vector was 0.008 ms, making it both fast and accurate enough for real-time BCI systems.

Chapter 3

Implementing a BCI with fuzzy logic

In order to demonstrate the usage of fuzzy classification algorithms, it was decided to implement a simple BCI application using EEG technology and fuzzy classification algorithms.

The application can be used to partially control the mouse cursor on a computer screen and to better illustrate the usefulness of fuzzy classifiers specifically, it is able to detect and carry out multiple directions simultaneously, which is not possible with traditional classification algorithms based on binary logic.

Firstly, the application collects training data from the user. The user is asked to give each of the three possible commands - "neutral", "right" and "up" for seven seconds ten times. This data is processed and then used to train train the classifier.

When the classifier is trained, the user is given the task of moving the computer cursor from the bottom left corner of a 200 by 200 pixel canvas, where it is initially placed to the top right corner, indicated by a green area (Figure 3.1).

The application records the data used to train the classifier, the data used to move the cursor and the cursor trajectory into comma-separated values (CSV) files and the trajectory of the cursor is saved as a bitmap image for later examination.

Ideally, the user would move the cursor in one straight diagonal line. With traditional classifiers trained for the same three mental tasks that would be impossible, as only one command would be recognised at a time - the cursor could only be moved in a series of up/right motions. We hope to show with experiments that using fuzzy classification allows us to move the cursor not only in the two trained directions but also diagonally.

The following sections describe the technology that was chosen for the implemen-

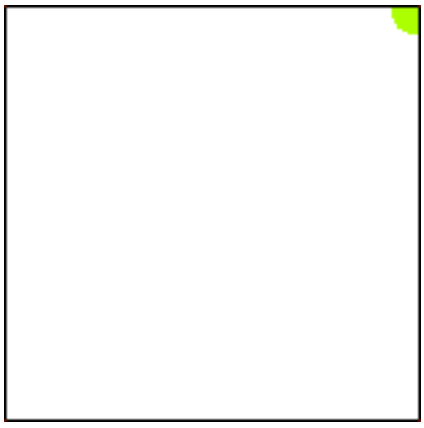

Figure 3.1: The user is tasked with moving the cursor from the bottom left to the top right corner of the canvas.

tation of the BCI and the process of implementation.

3.1 Choice of technology

The main part of the BCI is of course the EEG device. In this thesis, the Emotiv EPOC[sys] neuroheadset is used for this purpose.

Emotiv EPOC is a wireless neuroheadset that was developed by Emotiv Systems, an Australian electronics company, with the main purpose of being used as additional input for controlling computer games. It has 14 sensors for capturing electrical activity of the brain and muscles as well as a gyroscope for detecting the position of the user's head. For the purpose of this thesis, this headset is only used for recording EEG data and other sensors are ignored.

Due to the headset's low price, high portability and access to raw EEG data, it is useful for scientific EEG research.

The headset comes with proprietary software package for Microsoft Windows and Apple Mac OS X operating systems that allows for training of various actions, calibrating and monitoring the connectivity and signal quality of the headset and several versions of a Software Development Kit (SDK) are provided, but the easiest way of programmatically accessing raw EEG data is with Emokit[KM], an open source library for the C and Python programming languages that was built by reverse-engineering the encrypted protocol used by the EPOC headset.

For faster and easier prototyping and implementation of the BCI, the Python programming language was chosen. SciPy[JOP$^+$] and NumPy[Oli] libraries are used for processing the raw EEG data and Python Extensions for Windows[Ham] package is used to control the cursor on the Microsoft Windows platform.

The Emokit Python library has some issues with USB device connectivity in Apple Mac OS X operating system so for this application it was decided to focus on having the application working in Microsoft Windows operating system.

3.2 Implementation process

The implementation of the BCI consists of three main steps:

1. Capturing EEG data
2. Processing the data for machine learning
3. Training and classification

The Emokit library makes the first task very straightforward - the initialization process automatically detects the headset, connects to it and begins collecting packets of data. A process in a separate thread is used to read the packets out of Emokit's internal queue, filter out the unneeded data, leaving only the EEG readings and add the packet to the queue used by the rest of the application.

Each packet of data represents a single reading of all 14 sensors of the neuroheadset. By itself, this data isn't helpful enough to use it to directly train a classification algorithm. Instead, sets of successive packets are collected and processed as one instance of data.

The Emotiv EPOC headset produces a reading 128 times per second so it was decided to process each second separately. In order to treat a second (128 samples) of EEG data (readings from 14 sensors) as a single instance, the raw data must be transformed in a number of ways before it can be used to train the classification algorithm to produce meaningful results.

As described previously, electric activity of the brain is periodic in nature and the most popular way of analysing periodic signals is to apply the Fourier transform to transform the signal from time domain to frequency domain. Before transforming the signal, linear trends should be removed, because raw EEG data is noisy and the linear trend of the signal in time is of no interest to us. These transformations are made using the NumPy and SciPy Python libraries.

The Emotiv EPOC headset only captures electrical activity of frequency below 45Hz, so after the necessary transformations, the feature space consists of 630 features - 45 frequency components from each of the 14 sensors. For the training data, one additional field is added - the class of the instance.

This data can be used to train the classification algorithm.

First, an attempt was made to manually implement the naïve Bayes algorithm that would output the probability of all classes instead of just one. The algorithm worked as intended, but turned out to be too slow and not accurate enough for use in the application. It was decided to use a naïve Bayes classifier from the scikit-learn[PVG+11] Python library, which also had a method for outputting all the calculated probabilities.

The application was implemented so that both crisp and fuzzy classification can be used. When the crisp classification is being used, if the classifier labels an instance of data as "right", the cursor is moved 10 pixels to the right and if it is clasified as "up", the cursor is moved 10 pixels up.

In the case of fuzzy classification, probabilities for each class are calculated for each feature vector. If the probability of the "neutral" class is the highest, the cursor does not move, otherwise the cursor is moved to the right the amount of pixels that is the probability of the "right" class multiplied by 10 and up by the amount of pixels that is the probability of the "up" class multiplied by 10. For example, if the probability of both the "up" and "right" class is calculated as 0.5, the cursor is moved in a diagonal direction, 5 pixels to the right and 5 pixels up.

The experiment stops when the cursor reaches the top right corner of the canvas.

3.3 Results

Six experiments were performed with each of the five test subjects - three using crisp classification and three with using fuzzy classification. All the test subjects were briefly introduced to the technology and to what the aim of the experiment is and the difference between crisp and fuzzy classification was explained.

The test results for the subject 'at' are displayed in Figure 3.2. The figure shows the cursor trajectories of the three tests with crisp classification and the three tests with fuzzy classification. Time, in seconds, that it took the test subject to reach the goal is displayed under each test.

Similar figures for other test subjects can be viewed in Appendix A.

Our initial hopes were that the usage of the fuzzy classification algorithm in the BCI would allow the user to move the cursor in a diagonal direction, achieving a cursor trajectory closer to the ideal trajectory (a straight line from beginning to the end) than with crisp classification. The results clearly show that the movement during the tests with the fuzzy classification was mostly comprised of series of upward and rightward movements just like in the case of crisp classification and diagonal movement was only achieved a couple of times. To demonstrate the results in a more objective way, the area of the shape formed between the ideal trajectory and the actual achieved trajectory was measured. The results can be

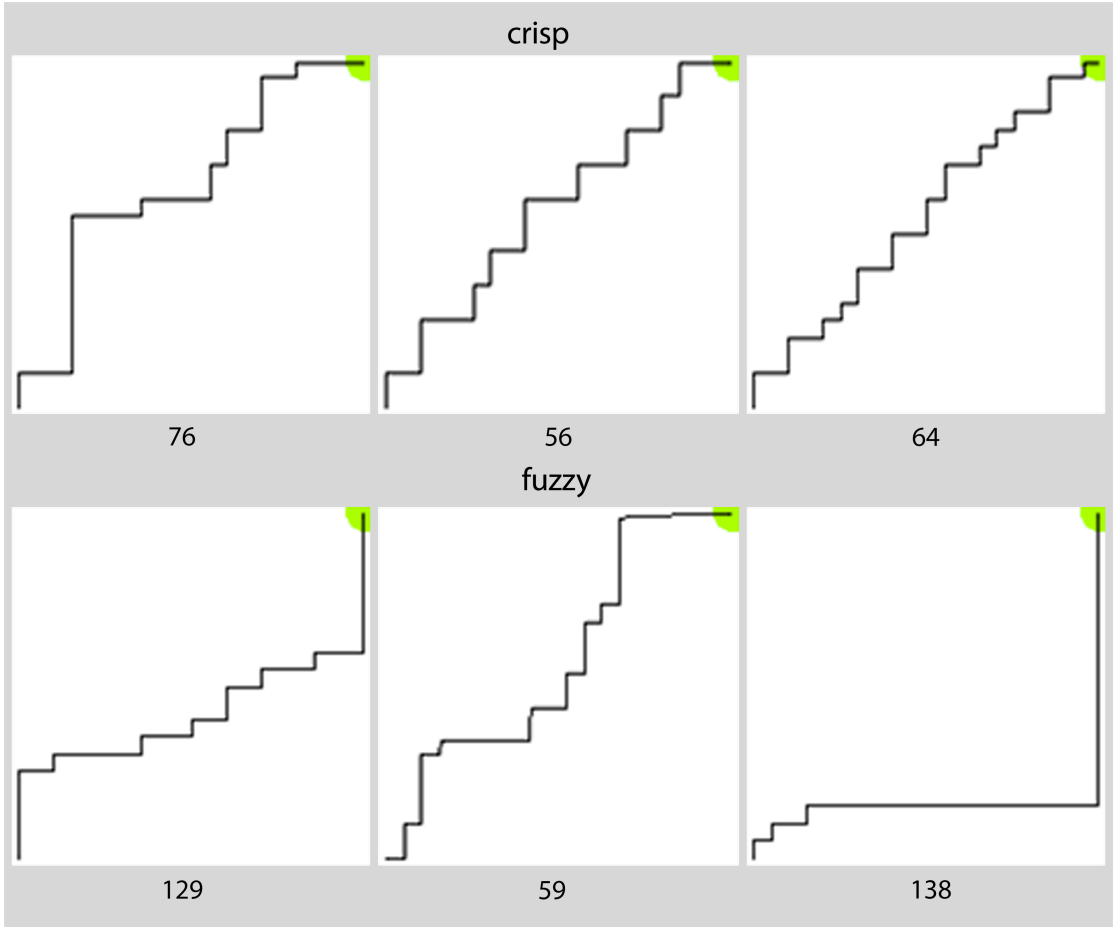

Figure 3.2: Test results for subject ‘at’

seen in Table 3.1.

A statistical test was performed on those numbers to determine whether the tests conducted with fuzzy classification produced a better trajectory than those with crisp classification.

The null hypothesis is that the average area of the shape produced by the trajectory of the cursor in an experiment with crisp classification is smaller or equal to that of an experiment with fuzzy classification and the alternative hypothesis is that the area is smaller in the case of fuzzy classification.

To test the hypothesis, we perform a one-tailed t-test with unequal variances. In our case, the z-statistic is 0.2176, which translates to a p-value of 0.4139, which is not significant at even a confidence level of 0.1. This shows that based on our data, we cannot prove that fuzzy classification provides significantly better results.

The test subjects themselves said that it was hard for them to come up with a set of mental commands that would move the cursor as desired in the case of fuzzy classification. Choosing two distinct commands for “up” and “right” was a much

Subject	Crisp (pixels)	Fuzzy (pixels)
tk	8986	7640
	1564	5074
	9604	6407
at	3279	3484
	1602	1928
	1009	7808
jm	4186	8254
	5995	3897
	10878	4820
mp	9131	4632
	6424	10878
	9993	5001
sb	4164	7476
	9575	1746
	4010	3249
Mean	6025,4667	5486,2667
Standard deviation	3330,8681	2467,6186

Table 3.1: Area of the shape formed between the ideal trajectory and the actual trajectory

more intuitive way of controlling the cursor than attempting to combine commands to perform a diagonal motion and simply thinking of an upwards, rightwards and diagonal motion didn't seem to yield the desired results.

Based on the results of the tests, we can say that there is no evidence to support the claim that fuzzy classification provides some sort of advantage over crisp classification in BCIs of this kind. However, there are still uncertainties in this method that could be further researched.

Other fuzzy classification methods, for example the fuzzy SVM, can be applied to EEG-based BCI and different ways of attempting to combine commands can be tested. A further research into the way such commands are detected would give more insight into the nature of the problem and perhaps show what sort of mental tasks are best suited for such applications.

Conclusion

This thesis examined the usage of fuzzy classification algorithms in brain-computer interfaces (BCI) based on electroencephalography (EEG). The goal was to see if fuzzy classification can be more efficient than crisp classification for the purpose of moving a cursor on a computer screen with a BCI because it can detect multiple mental commands simultaneously and move the cursor in a diagonal direction when the classifier is only trained with horisontal and vertical mental tasks.

The existing literature concerning the use of traditional crisp classifiers in BCI, the research of fuzzy classification and its use in BCI was examined.

A simple BCI application using fuzzified Naïve Bayes was implemented and a series of experiments were carried out in which test subjects were tasked with moving a cursor from the bottom left corner of a canvas to the upper right corner. The test results show that there is no evidence to support the claim that a fuzzy classifier is better, more efficient or more intuitive than a crisp classifier in this kind of applications. In fact, most test subjects found it hard to come up with appropriate mental commands for such an application.

Bibliography

[AI02] Shigeo Abe and Takuya Inoue. Fuzzy support vector machines for multiclass problems. In *ESANN*, pages 113–118, 2002.

[AKM05] Bassel Abou-Khalil and Karl E Misulis. *Atlas of EEG & Seizure Semiology: Text with DVD*. Butterworth-Heinemann, 2005.

[BC00] Kristin P Bennett and Colin Campbell. Support vector machines: hype or hallelujah? *ACM SIGKDD Explorations Newsletter*, 2(2):1–13, 2000.

[Bis95] Christopher M Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

[BMBB04] Jaimie F Borisoff, Steven G Mason, Ali Bashashati, and Gary E Birch. Brain-computer interface design for asynchronous control applications: improvements to the lf-asd asynchronous brain switch. *Biomedical Engineering, IEEE Transactions on*, 51(6):985–992, 2004.

[CGM+92] Gail A Carpenter, Stephen Grossberg, Natalya Markuzon, John H Reynolds, and David B Rosen. Fuzzy artmap: A neural network architecture for incremental supervised learning of analog multidimensional maps. *Neural Networks, IEEE Transactions on*, 3(5):698–713, 1992.

[CGR91a] Gail A Carpenter, Stephen Grossberg, and John H Reynolds. Artmap: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. *Neural networks*, 4(5):565–588, 1991.

[CGR91b] Gail A Carpenter, Stephen Grossberg, and David B Rosen. Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural networks*, 4(6):759–771, 1991.

[CMP08] Damien Coyle, T Martin McGinnity, and Girijesh Prasad. A multiclass brain-computer interface with sofnn-based prediction preprocessing. In *Neural Networks, 2008. IJCNN 2008.(IEEE World*

Congress on Computational Intelligence). IEEE International Joint Conference on, pages 3696–3703. IEEE, 2008.

[CPM09] Damien Coyle, Girijesh Prasad, and T Martin McGinnity. Faster self-organizing fuzzy neural network training and a hyperparameter analysis for a brain–computer interface. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 39(6):1458–1471, 2009.

[CST+03] F Cincotti, A Scipione, A Timperi, D Mattia, MG Marciani, J Millan, S Salinari, L Bianchi, and F Bablioni. Comparison of different feature classifiers for brain computer interfaces. In *Neural Engineering, 2003. Conference Proceedings. First International IEEE EMBS Conference on*, pages 645–647. IEEE, 2003.

[dRMMF+02] J del R Millan, Josep Mouriño, Marco Franzé, Febo Cincotti, Markus Varsta, Jukka Heikkonen, and Fabio Babiloni. A local neural classifier for the recognition of eeg patterns associated to mental tasks. *Neural Networks, IEEE Transactions on*, 13(3):678–686, 2002.

[eeg13] Eeg: Medlineplus medical encyclopedia, 2013. `http://www.nlm.nih.gov/medlineplus/ency/article/003931.htm`.

[FS91] Peter W Frey and David J Slate. Letter recognition using holland-style adaptive classifiers. *Machine Learning*, 6(2):161–182, 1991.

[GEV03] Gary N Garcia, Touradj Ebrahimi, and Jean-Marc Vesin. Support vector eeg classification in the fourier and time-frequency correlation domains. In *Proc. 1st IEEE-EMBS Conf. on Neural Engineering (Capri Island, Italy)*, pages 591–4, 2003.

[Gro03] Stephen Grossberg. *Adaptive resonance theory*. Wiley Online Library, 2003.

[Ham] Mark Hammond. pywin32, python extensions for windows. `http://sourceforge.net/projects/pywin32/`.

[Has88] A Hashizume. Fully automated blood cell differential system and its application. In *Proc. IUPAC 3rd International Congress on Automation and New Technology in the Clinical Laboratory*, pages 297–302, 1988.

[HL98] Tzung-Pei Hong and Chai-Ying Lee. Learning fuzzy knowledge from training examples. In *Proceedings of the seventh international conference on Information and knowledge management*, pages 161–166. ACM, 1998.

[IA01] Takuya Inoue and Shigeo Abe. Fuzzy support vector machines for pattern classification. In *Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on*, volume 2, pages 1449–1454. IEEE, 2001.

[IHW11] Mark A. Hall Ian H. Witten, Eibe Frank. *Data Mining Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers, 2011.

[JOP+] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. `http://www.scipy.org/`.

[KC94] Hon Keung Kwan and Yaling Cai. A fuzzy neural network and its application to pattern recognition. *Fuzzy Systems, IEEE Transactions on*, 2(3):185–193, 1994.

[KGG85] James M Keller, Michael R Gray, and James A Givens. A fuzzy k-nearest neighbor algorithm. *Systems, Man and Cybernetics, IEEE Transactions on*, (4):580–585, 1985.

[KM] Cody Brocious Kyle Machulis. Emokit. `https://github.com/qdot/emokit`.

[LCL+07] Fabien Lotte, Marco Congedo, Anatole Lécuyer, Fabrice Lamarche, Bruno Arnaldi, et al. A review of classification algorithms for eeg-based brain–computer interfaces. *Journal of neural engineering*, 4, 2007.

[Lot06] Fabien Lotte. The use of fuzzy inference systems for classification in eeg-based brain-computer interfaces. In *3rd International Brain-Computer Interfaces Workshop and Training Course*, 2006.

[LW02] Chun-Fu Lin and Sheng-De Wang. Fuzzy support vector machines. *Neural Networks, IEEE Transactions on*, 13(2):464–471, 2002.

[MW05] Dennis J McFarland and Jonathan R Wolpaw. Sensorimotor rhythm-based brain-computer interface (bci): feature selection by regression improves performance. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 13(3):372–379, 2005.

[Oli] Travis Oliphant. Numpy, 2006–. `http://www.numpy.org`.

[PM99] Irina Perfilieva and Jiří Močkoř. *Mathematical principles of fuzzy logic*. Springer, 1999.

[PPNS02] Ramaswamy Palaniappan, Raveendran Paramesran, Shogo Nishida, and Naoki Saiwaki. A new brain-computer interface design using fuzzy artmap. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 10(3):140–148, 2002.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[Rab89] Lawrence Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[RJ91] Sarunas J Raudys and Anil K. Jain. Small sample size effects in statistical pattern recognition: Recommendations for practitioners. *IEEE Transactions on pattern analysis and machine intelligence*, 13(3):252–264, 1991.

[Sar95] Warren S Sarle. Why statisticians should not fart. *Cary, NC, USA*, 1995.

[sys] Emotiv systems. Epoc neuroheadset. `http://www.emotiv.com/apps/epoc/299/`.

[TAT+91] Hiroshi Takenaga, Shigeo Abe, Masao Takato, Masahiro Kayama, Tadaaki Kitamura, and Yoshiyuki Okuyama. Input layer optimization of neural networks by sensitivity analysis and its application to recognition of numerals. *Electrical Engineering in Japan*, 111(4):130–138, 1991.

[TPLX02] Yongchuan Tang, Wuming Pan, Haiming Li, and Yang Xu. Fuzzy naive bayes classifier based on fuzzy clustering. In *Systems, Man and Cybernetics, 2002 IEEE International Conference on*, volume 5, pages 6–pp. IEEE, 2002.

[WC07] Tai-Yue Wang and Huei-Min Chiang. Fuzzy support vector machine for multi-class text categorization. *Information Processing & Management*, 43(4):914–929, 2007.

[WK90] Sholom M Weiss and Ioannis Kapouleas. An empirical comparison of pattern recognition, neural nets and machine learning classification methods. *Readings in machine learning*, pages 177–183, 1990.

Internet URLs were valid on 14.05.2014

Appendix A
Test results

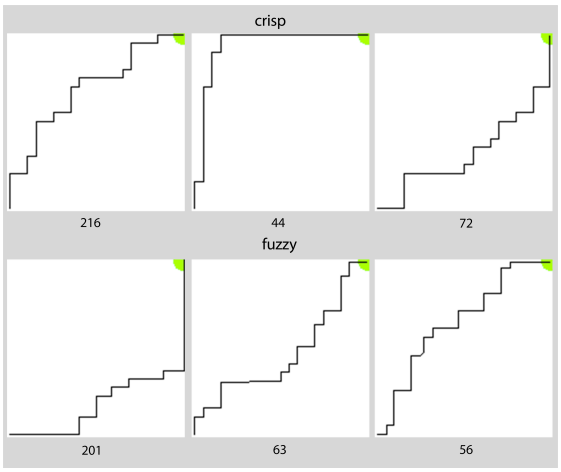

Figure 1: Test results for the subject ‘sb’

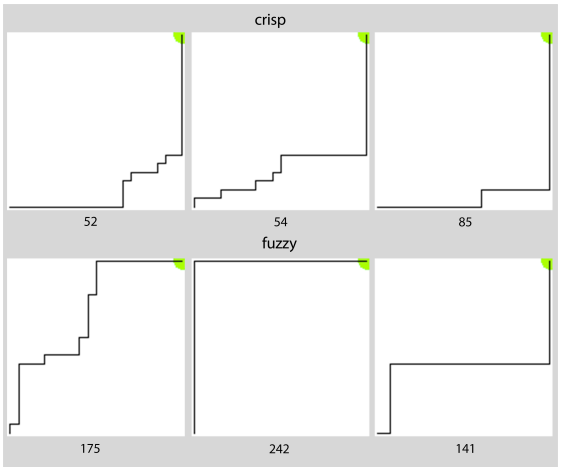

Figure 2: Test results for the subject ‘mp’

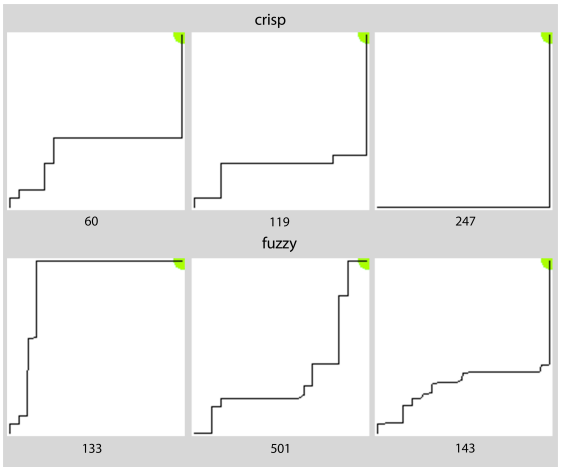

Figure 3: Test results for the subject ‘jm’

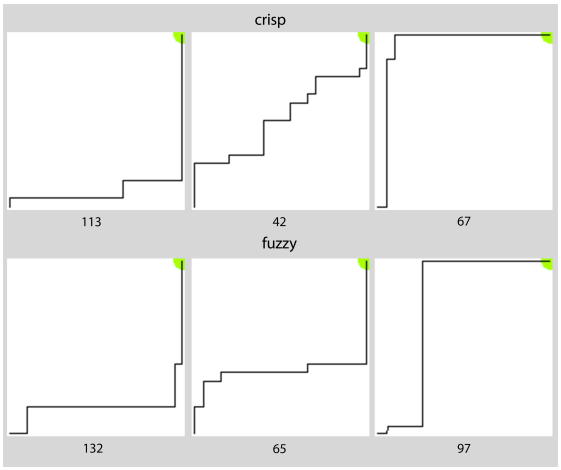

Figure 4: Test results for the subject ‘tk’

License

Non-exclusive license to reproduce thesis and make thesis public

I, Stepan Bolotnikov (date of birth: 21.02.1992),

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

   "Usage of fuzzy classification algorithms in brain-computer interfaces", supervised by Ilya Kuzovkin,

2. am aware of the fact that the author retains these rights.

3. certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 14.05.2014