Investigating the use of multi-label classification methods for the purpose of classifying electromyographic signals

Petter Björklund

Master's thesis in
Biomedical Engineering

Department of Biomedical Engineering
Faculty of Engineering
Lund University

Supervisors: Christian Antfolk and Nebojsa Malesevic

Januari 2018

Abstract

The type of pattern recognition methods used for controlling modern prosthetics, referred to here as single-label classification methods, restricts users to a small amount of movements. One prominent reason for this is that the accuracy of these classification methods decreases as the number of allowed movements is increased. In this work a possible solution to this problem is presented by looking into the use of multi-label classification for classifying electromyographic signals. This was accomplished by going through the process of recording, processing, and classifying electromyographic data. In order to compare the performance of multi-label methods to that of single-label methods four classification methods from each category were selected. Both categories were then tested on their ability to classify finger flexion movements. The most commonly tested set of movements were the thumb, index, long, and ring finger movements in addition to all the possible combinations of these four fingers. The two categories were also tested on their ability to learn finger combination movements when only individual finger movements were used as training data. The results show that the tested single- and multi-label methods obtain similar classification accuracy when the training data consists of both individual finger movements and finger combination movements. The results also show that none of the tested single-label methods and only one of the tested multi-label methods, multi-label rbf neural networks, manages to learn finger combination movements when trained on only individual finger movements.

Preface

This master thesis was performed between January and December of 2017 at the department of biomedical engineering at Lund University. The master thesis was supervised by Christian Antfolk acting as main supervisor and Nebojsa Malesevic acting as assistant supervisor. The examiner of this master thesis was Johan Nilsson.

Acknowledgments

This project would not have been possible without the support of several people. First I would like to thank Christian Antfolk for the help he provided and for coming up with the idea of the thesis. I would also like to thank Nebojsa Malesvic for all the help and guidance he provided during this entire process. Finally I would like to thank my family and friends for their support and for always pushing me forward.

Contents

1 Introduction

The loss of a limb is an intensely traumatic experience that almost always leads to a noticeable decrease in a persons perceived quality of life [5]. A part of this decrease comes from the forced life style change that limb loss imposes upon a person. In order to help these people prosthetic devices were invented to compensate for the missing limb. Over the years the quality of these devices have changed remarkably from the primitive peg-leg and metal arm to the more complicated myoelectric-controlled prosthesis that exist today which is illustrated in figure 1. Yet even with today's technology a significant amount of people chose to forego the more advanced prosthetic devices in favor of static or body powered prostheses or even no prostheses at all [7]. There are a wide variety of reasons for this choice ranging from cumbersome weight to lack of feedback to poor control etc.
In this work the problem of interest lies within the control of prosthetic devices and more specifically in the control of myoelectric hand prostheses. A myoelectric-controlled hand prosthesis works by measuring the electrical activity of the nerves in the residual limb. There are different ways that a hand prosthesis can be connected to a users body. The most common way is that the residual arm is placed into a socket in the prosthesis which is then fastened around the arm. This socket is filled with electrodes which allows for measurement of the nerve activity. Another much rarer way is through a procedure known as osseointegration in which the prosthesis is connected directly to the bone which can allow the electrodes to be directly connected to the users muscle tissue.

Controlling a hand prosthesis is done through a series of steps. The user first decides what sort of movement should be made which for myoelectric prostheses is done by contracting muscles in the residual arm. The electrodes in the prosthesis detect the nerve activity due to the muscle contraction and then runs the detected signal through a pattern recognition algorithm. The output from this algorithm then decides what type of movement the prosthesis performs. The movement that is performed is all or nothing so a user can neither stop the movement midway nor can another movement be made until the current one is completed.
One of the most evident problems with the current pattern recognition methods used for controlling hand prostheses is that increasing the amount of possible hand and finger movements has a direct impact on the classification accuracy [10]. This means that restrictions have to be placed on how many

Figure 1: (Left) Old iron arm prosthesis believed to be dated from 1560-1600 [19]. (Right) Michelangelo prosthetic hand developed by Ottobock [21]

degrees of freedom that are allowed which has negative impact on user satisfaction. Another problem that occurs when increasing the degrees of freedom is that the training time required increases greatly since each movement has to be trained individually.

A potential solution to these problems could be to change the perspective. Instead of looking at each movement as an individual class separate from all others it can instead be seen as combination of different classes. This can be done by considering each finger as its own class which are referred to as labels so that a closed hand for example could correspond to all labels being active while an open hand would have zero labels active. These types of classification methods that see each class as a set of labels are called multi-label classification methods. A point of interest with multi-label classification is the possibility that training might only be needed on individual labels without any need for training on the label combinations.

Previously multi-label classification methods have been used with success in areas like text categorization [15] and image classification [3] but it still remains untested for time signals like electromyographic signals.

1.1 Aim

The aims of this thesis are as follows:

- To test the possibility of applying multi-label classification methods for the purpose of classifying electromyographic signals.
- To compare the performance of multi-label methods with that of single-label methods when it comes to classifying electromyographic signals.
- To test the ability of multi-label classification methods to extrapolate from the training on individual labels in order to classify label combinations.

1.2 Disposition

This report contains 6 chapters. Chapter 1 outlines why the study has been done and what the goal of it was. Chapter 2 discusses the theory that the study is built upon. Chapter 3 explains the methods used in this work both in what was done and how it was done. Chapter 4 contains the results of the study. Explanations and theories for the results are written in chapter 5. Finally in chapter 6 the conclusion is stated and some avenues for further work are explored.

2 Theory

In this section the theory needed to understand the processes required for multi-label classification is explained. The steps needed in order to perform a classification are illustrated in figure 2. Section 2.1 will discuss why feature extraction is required for functional classification and will also explain the different features used in this work. Section 2.2 and 2.3 will discuss the different types of classifiers used in this work. Finally in section 2.4 some ways of measuring error in multi-label classification are explained. The recording and data processing steps are discussed in section 3.

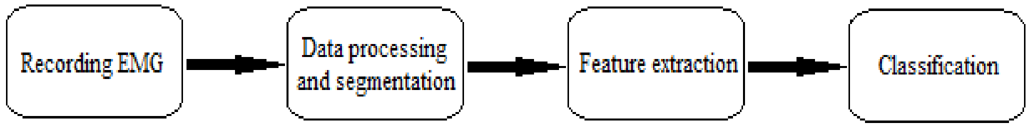

Figure 2: Block diagram of the steps involved in attempting to classify EMG signals

2.1 Features

Electromyograhpic (EMG) signals contain a lot of noise from different sources [4] which means that trying to classify raw EMG data often results in a poor classification accuracy. In order to handle this problem a method called feature extraction can be used on the EMG signals to extract useful information hidden within the data. Feature extraction works by letting a window slide over the signal and then calculating the desired feature on the windowed segment. These feature then form an N dimensional space where N is the number of features used times the number of channels of EMG data. The idea is then that by using different features the separation between classes in this N dimensional space is increased which should make classification easier. Thus by letting the extracted features be the input into the classifier instead of the raw EMG data the classification accuracy can be improved [4]. The features used for EMG signals can be divided into four categories: time domain, frequency domain, time-frequency domain and spatial domain [17]. In this work only two of the four feature domains were used, time domain and frequency domain.

2.1.1 Time Domain Features

Time domain features tend to be the easiest to calculate compared to other feature categories due to being calculated from the raw EMG signal without a need for any transformation. In addition time domain features also maintain high classification accuracy in comparison to other feature categories [23]. The time domain features used in this work are as follows:

Mean Absolute Value

This feature simply calculates the mean absolute value (MAV) of the windowed signal. MAV is one of the more popular time domain features and is often used for the detection of muscle contraction levels [22]. The MAV can be calculated by:

$$MAV = \frac{1}{N}\sum_{n=1}^{N} |x_n| \tag{1}$$

Where N is the total number of samples in the window and x_n represents the n^{th} sample in the window.

Variance

The variance (VAR) of a EMG signal gives a measure of the signals power. The variance can be expressed as:

$$VAR = \frac{1}{N-1}\sum_{n=1}^{N} x_n^2 \tag{2}$$

Zero Crossings

This feature gives the number of times that the signal, $\mathbf{x}$, crosses zero within the window. This feature is very sensitive to noise so a threshold is often needed. Zero Crossings (ZC) is related to the frequency of the EMG-signal. The ZC value increases by one step if the following is true:

$$\begin{aligned} &(\ \{x_n > 0 \ and \ x_{n+1} < 0\} \ or \ \{x_n < 0 \ and \ x_{n+1} > 0\}\ ) \\ &and \ |x_n - x_{n+1}| \geq \epsilon \end{aligned} \tag{3}$$

Where ϵ is the chosen threshold value.

Slope Sign Change

This feature gives the amount of times that the slope of the signal $\mathbf{x}$ changes sign within the window, it is related to the frequency of the signal. Slope sign change (SSC) is quite similar to ZC and as with ZC it is sensitive to noise so a threshold is often necessary. The SSC value is incremented by one if the following is true:

$$\begin{aligned}&(\{x_n > x_{n-1} \textit{ and } x_n > x_{n+1}\} \textit{ or } \{x_n < x_{n-1} \textit{ and } x_n < x_{n+1}\})\\ &\textit{and } (|x_n - x_{n+1}| \geq \epsilon \textit{ or } |x_n - x_{n-1}| \geq \epsilon)\end{aligned} \tag{4}$$

Where ϵ is the threshold.

Waveform Length

The waveform length (WL) feature gives a measure of the amplitude, frequency, and time of the signal. The WL is calculated as the summed up length of the signal within the window:

$$WL = \sum_{n=1}^{N} |x_{n+1} - x_n| \tag{5}$$

Willison Amplitude

The Willison amplitude (wAmp) gives the amount of times that the amplitude of the signal has changed more than a set threshold between two time instances. The wAmp is related to the firing of motor unit action potentials and thus gives a measure of the muscle contraction level [22].

$$\begin{aligned}\text{wAmp} &= \sum_{n=1}^{N} f(|x_n - x_{n+1}|),\\ f(x) &= \begin{cases} 1 \text{ if } x > \text{threshold} \\ 0 \text{ otherwise}\end{cases}\end{aligned} \tag{6}$$

Log detector

The log detector (logDet) feature gives an indication of muscle contraction force [16]. This feature can be calculated using:

$$\text{logDet} = e^{\frac{1}{N}\sum_{n=1}^{N} \log|x_n|} \tag{7}$$

2.1.2 Frequency Domain Features

When it comes to EMG signals frequency domain features are commonly used for measuring muscle fatigue or changes in motor unit recruitment and firing patterns [11]. They take a longer time to calculate compared to time domain features since the EMG signal has to be transformed in to frequency domain before the features can be calculated. The frequency domain features used in this work are:

Mean Frequency

This feature gives the mean frequency (FMN) of the power spectral density of the EMG signal. The FMN is calculated by:

$$FMN = \frac{\sum_{n=1}^{N} f_n P_n}{\sum_{n=1}^{N} P_n} \tag{8}$$

Where f_n is the frequency of the spectrum, P_n is the EMG power spectrum and N gives the size of the frequency bin.

Median Frequency

The median frequency (FMD) feature gives the frequency at which the spectrum is divided into two equally large parts. The FMD frequency is given by:

$$FMD = \frac{1}{2}\sum_{n=1}^{N} P_n \tag{9}$$

Peak Frequency

The peak frequency (PKF) feature gives the frequency at the point where maximum power occurs. PKF can be expressed as:

$$PKF = \max P_n, \text{ where } n = 1, ..., N \tag{10}$$

2.2 Single-label Classification

Classification is the process of assigning class labels to an observation based on a model created by a set of training data. The most common type of classification is single-label classification which occurs when each observation is restricted to having only one class label. single-label classification sees use in an incredibly varied amount of fields with some examples being speech recognition [6], geostatistics [18] and credit scoring [12] . There are many different methods for implementing single-label classification each with its own advantages and drawbacks, the methods relevant for this work are described below.

2.2.1 K-Nearest Neighbour

The k-nearest neighbour (KNN) algorithm is one of the more simplistic classification methods. KNN works on the idea that observations with the same class label are grouped together. This means that if a new observation lies close to a majority of instances with a specific label then the observation will be assigned the same class label as the majority. Calculating the distance between the observations can be done in different ways with some examples being Euclidean distance, Chebychev distance and Hamming distance. The advantages of KNN lies in how easy it is to implement and having average accuracy despite its simplicity [13]. In comparison a notable disadvantage of the KNN method is that the entire training set is saved and then used as a sort of look-up table. This means that if the training set is large the KNN method will take a longer time to compute. An example of how the KNN algorithm works can be seen in figure. 3.

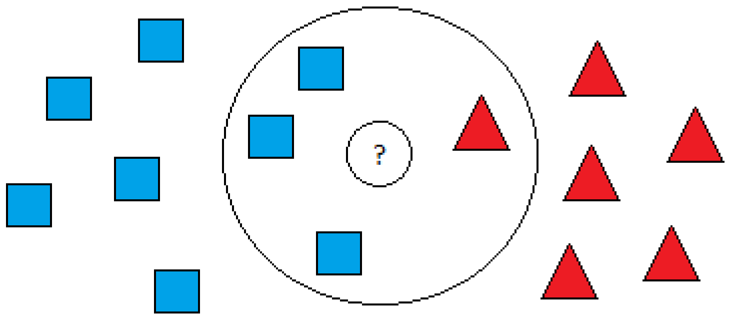

Figure 3: Example of k-nearest neighbour with $k = 4$. The blue squares outnumber the red triangles within the neighbourhood so the unknown observation is given the blue square class label.

2.2.2 Naive Bayes

Naive Bayes (NB) is a type of probabilistic classification method. The method gets its name from its use of the Bayes theorem. The naive part comes from the assumption that the predictor variables are independent from each other. The method works by assuming that the observations are samples from probability distributions with each class having its own distribution. By calculating these probability distributions it would be possible to determine the probability that a new observations happens at a specific point given that it belongs to a specific class. The Bayes theorem can then be used to instead calculate the probability that a new observation belongs to a specific class given that it occurs at a given point. The classification is then done by calculating the probability of a new observation belonging to each class and then selecting the class with the highest probability.

2.2.3 Support Vector Machine

A Support Vector Machine (SVM) is a classification method that functions by finding the hyper plane that linearly separates all data points as much as possible. This is done by trying to maximize the margin which is the the distance from the boundary between the classes and the observations closest to it under the constraint that all observations are on the correct side of the boundary. Regrettably most data is not linearly separable especially not if any type of noise is involved. One way to handle this problem is to allow observations to be on the wrong side of the boundary and then add a penalty for each observation on the wrong side. This directs the method towards maximizing the margin while trying to keep misplaced observations to a minimum. But even if this is done there is still no guarantee that the data can be linearly separated instead it might only be possible by using a nonlinear boundary. The problem with nonlinear boundaries is that they are far more difficult to calculate. A way to solve this issue is by making use of the kernel trick [2]. In short the kernel trick works by mapping the data to different and possibly higher dimensional space where a hyper plane that separates the classes can be more easily found. The linear boundary that is found in the mapped space is then transformed into a nonlinear boundary when the data is mapped back into the original space. For the case where more then 2 classes are involved the SVM method combines multiple binary classifiers in order create boundaries that separate all the classes. An illustration of the

SVM method can be seen in figure 4.

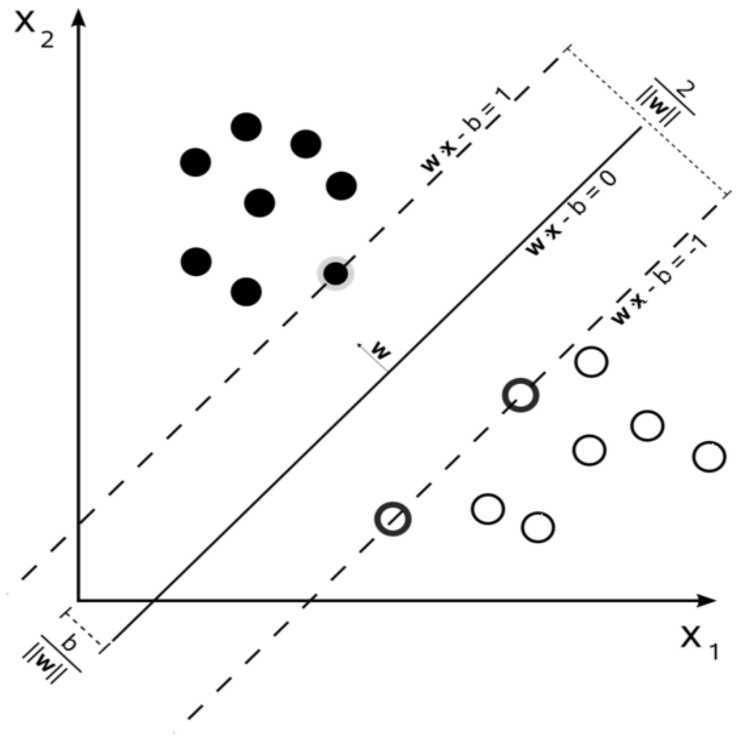

Figure 4: Maximum-margin hyper plane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors.

2.2.4 Linear Discriminant Analysis

Linear discriminant analysis (LDA) is a commonly used classification method. LDA works by creating a boundary between the classes so that a new observation can be classified by simply looking at where it occurs in relation to the boundaries. In order to create these boundaries the observations are treated as samples from multivariate normal distribution functions with each class having its own distribution function. The boundaries can then be found by looking at the set of points where the probability that an observation belongs to two classes is equal. This can be accomplished by making use of the vector of the mean values and the covariance matrix of each distribution. By making the assumption that the covariance matrices of the distributions are equal the calculations become simpler and the boundaries between the classes become linear. If the assumption of equal covariance matrices is not made the boundaries become quadratic and the method instead turns into quadratic discriminant analysis.

2.2.5 Neural Networks

This section serves only as an introduction into Neural Networks (NN) for single-label classification so that it can later be expanded into multi-label classification in section 2.3.4. NN are a set of different classification methods that were inspired from the interaction between neurons inside the brain. Similar to a real brain NN consist of a number of connected neurons. These neurons are structured into three different layers, the input layer, the hidden layer, and the output layer as can be seen in figure 5. Each neuron has an activation function that determines whether or not it will pass on information to the subsequent layer. There are many different activation function that can be used but in this work only the radial basis functions such as the Gaussian function are of interest. The input to each neuron is calculated by multiplying the output of all the connected neurons from the previous layer with appropriate weights then summing them plus a bias term together and then putting it through the neurons activation function. The output of a neuron then depends on its input, its activation function and its output function which is often chosen to be the identity function. For single-label classification the inputs in the input layer correspond to the predictors so if 5 predictors are used the input layer will contain 5 neurons. In the output layer each output is connected to a class and the values from each output gives the probability that an observation belongs to that class. The classification is then completed by selecting the class with the highest output.

2.3 Multi-label Classification

There are many cases were an observation can have several different class labels that fit equally well. An image for example could contain both a mountain, a lake and a forest. If a single-label method was used to classify this image only one of these labels could be attached to the observation and information would be lost. To solve these kinds of problems multi-label classification methods can be used instead. In multi-label classification an observation can belong to several different classes. This means that each observation has a corresponding label set that says which labels the observation

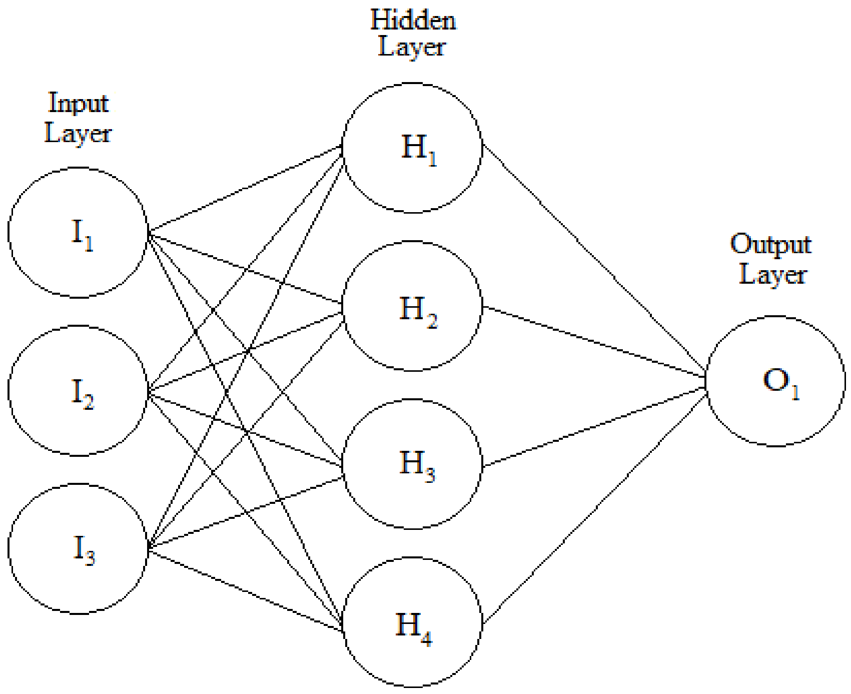

Figure 5: Illustration of a neural network with 3 inputs, 1 output and a hidden layer of width 1 and height 4.

has been fitted with.
There are two main areas of multi-label classification namely problem transformation and algorithm adaption [26]. In problem transformation the goal is to fit the data to the method. One example of how this can be done is by looking at each possibly label combination as its own class the data is then transformed from multi-label to single-label. The data can then be classified by using an already known single-label classification method. In algorithm adaption on the other hand the goal is to fit the method to the data. This has lead to the creation of methods that can handle multi-label data many of which are based on single-label methods. In this work only methods that belong to the algorithm adaption group are of interest. The methods that are used in this work can found below.
Multi-label classification has become more popular as the complexity of modern applications increases [26]. These days multi-label classification methods have been used with success among such varied fields as text categorization

[15], image classification [3] and protein function classification [20].

2.3.1 Multi-label K-Nearest Neighbour

Multi-label k-nearest neighbour (MLKNN) created by Min-Ling Zhang and Zhi-Hua Zhou [30] is a modification to the classic KNN method meant to allow for a lazy learning approach to multi-label data. The explanation of the MLKNN method is paraphrased from [30] in the following text.
Before starting the explanation of how the MLKNN method works some notation first needs to be presented. The observation to be classified is given as x and its corresponding label set is given as $Y \subseteq L$ where L is the set of all labels. A category vector for x is then defined $\vec{y}_x$ which gives whether or not a label is part of the observations label set:

$$\vec{y}_x(l) = \begin{cases} 1 \text{ if } l \in Y \\ 0 \text{ otherwise} \end{cases}, \; l \in L \tag{11}$$

This means that the purpose of the MLKNN method is to calculate the category vector since that in turn gives the label set of the observation. Another variable $N(x)$ can then be defined which gives the set of KNNs of x that were identified in the training set. By looking at the label set of each of these neighbours a membership counting vector can be defined:

$$\vec{C}_x(l) = \sum_{a \in N(x)} \vec{y}_a(l), \; l \in L \tag{12}$$

Where $\vec{C}_x(l)$ measures the number of neighbours of x that contain l in their label sets. Next H_1^l is defined as the event in which the observation x has label l in its label set while H_0^l is defined as the event where it does not. Finally E_j^l $(j \in 0, 1, ..., K)$ is defined as the event in which the sum of the occurrences of label l in the neighbours of observation x is equal to j. Now that the notations have been introduced the explanation of MLKNN can begin. The MLKNN method starts by identifying the KNNs in the training set for the input observation x. Once this is done the category vector can then be calculated with the help of the membership counting vector and the maximum a posteriori (MAP) principle:

$$\vec{y}_x(l) = \underset{b \in \{0,1\}}{\arg\max} \, P(H_b^l | E^l_{\vec{C}_x(l)}), \quad l \in L \tag{13}$$

This equation can then be rewritten by making use of the Bayes theorem:

$$\begin{aligned}\vec{y}_x(l) &= \underset{b\in\{0,1\}}{\arg\max}\ \frac{P(E^l_{\vec{C}_x(l)}|H^l_b)P(H^l_b)}{P(E^l_{\vec{C}_x(l)})} \\ &= \underset{b\in\{0,1\}}{\arg\max}\ P(E^l_{\vec{C}_x(l)}|H^l_b)P(H^l_b), \quad l \in L\end{aligned} \tag{14}$$

This means that the category vector and thus the label set of the observation can be calculated by only knowing the prior and posterior probabilities. These probabilities can in turn be estimated from the training set by counting the frequency of appearance of the different labels.

2.3.2 Multi-label Naive Bayes

Multi-label naive Bayes (MLNB) created by Min-Ling Zhang, José M. Peña, and Victor Robles [29] is a modification to the naive Bayes single-label classification method. Like the single-label version MLNB keeps the assumption of class conditional independence. The explanation of the MLNB method is paraphrased from [29] in the following text.
The same notation as that in the previous section is used here with the addition that the observation x is written out as $\mathbf{x} = (x_1, ..., x_d)$ where d is the total amount of predictors in the observation. The category vector can then be calculated with the same MAP principle that was used in section 2.3.1:

$$\vec{y}_x(l) = \underset{b\in\{0,1\}}{\arg\max}\ P(H^l_b|\mathbf{x}), \quad l \in L \tag{15}$$

By making use of the Bayes theorem this equation can be rewritten into:

$$\vec{y}_x(l) = \underset{b\in\{0,1\}}{\arg\max}\ \frac{P(\mathbf{x}|H^l_b)P(H^l_b)}{P(\mathbf{x})} = \underset{b\in\{0,1\}}{\arg\max}\ P(H^l_b)\prod_{i=1}^{d} P(x_i|H^l_b), \quad l \in L \tag{16}$$

The H^l_b probability can be calculated from the frequency of the labels in the training data while the $P(t_i|H^l_b)$ term can be calculated from the following Gaussian probability density function:

$$P(x_i|H^l_b) = \frac{1}{\sqrt{2\pi\sigma_i^{lb^2}}}\exp\Big(-\frac{(x_i-\mu_i^{lb})^2}{2\sigma_i^{lb^2}}\Big),\ 1 \le i \le d \tag{17}$$

But in order to solve this equation μ and σ first have to be calculated. This can be done by first defining the set $V = \{x_{nk} | \vec{y}_x(l) == b\}, 1 \leq n \leq M$ and then using this to calculate both μ and σ:

$$\mu_i^{lb} = \frac{1}{|V|} \sum_{v \in V} v \tag{18}$$

$$\sigma_i^{lb} = \sqrt{\frac{1}{|V| - 1} \sum_{v \in V} (v - \mu_i^{lb})^2} \tag{19}$$

Once this has been done the probabilities can be solved for and then used in order to determine the category vector.

2.3.3 Rank Support Vector Machines

Rank Support Vector Machines (Rank-SVM), created by André Elisseeff and Jason Weston, is a ranking based SVM method that allows for SVM methods to be extended into the realm of multi-label classification problems. The explanation for the Rank-SVM method is paraphrased from [9] and [8] in the following text.
The same notation that has been used in the previous sections is also used in this one but with the added definition of the training data. The training data is expressed as $D = \{(\mathbf{x}_i, Y_i) | 1 \leq i \leq m\}$ where m is the total amount of training instances.
The goal of the Rank-SVM method is to find a function f that minimizes the generalization error:

$$R(f) = E_{(\mathbf{x},Y)}[c(f, \mathbf{x}, Y] \tag{20}$$

Where c is a loss function with a structure dependant on how f is computed. This is done by first creating a ranking function that ranks the labels for each input. A threshold can then be defined where labels that are ranked higher than this are placed in the label set while those that are ranked lower are not. The rank of a label is defined as

$$r_l(\mathbf{x}) = \langle w_l, \mathbf{x} \rangle + b_l, \ l \in L \tag{21}$$

Where w_l is a weight vector and b_l is a bias. Thus a label l is part of the label set Y if $r_l(\mathbf{x}) > t(\mathbf{x})$ where $t(\mathbf{x})$ is the previously mentioned threshold.

Since the labels are to be sorted through a ranking system the concept of ranking loss becomes important. The ranking loss gives the average fraction of pairs that are incorrectly ordered.

$$RL = \frac{1}{|Y||\bar{Y}|}|(i,j) \in Y * \bar{Y} \text{ such that } r_i(x) \leq r_j(x)| \tag{22}$$

Where $\bar{Y}$ is the complementary set of Y.
As with single-label SVM method in Rank-SVM one of the goals is to maximize the margin discussed in 2.2.3. The margin of $(\mathbf{x}, Y)$ can be expressed as:

$$\min_{l \in Y, k \in \bar{Y}} \frac{\langle w_l - w_k, \mathbf{x}\rangle + b_l - b_k}{||w_l - w_k||}, \; Y \in L, \; \bar{Y} \notin L \tag{23}$$

Which gives the signed euclidean distance of $\mathbf{x}$ to the decision boundary. The w_l parameters can be normalized such that $\langle w_l - w_k, \mathbf{x}\rangle + b_l - b_k \geq 1$ with equality for some $\mathbf{x} \in D$ and $l, k \in Y * \bar{Y}$. With some additional calculations that are detailed in [9] a way to maximize the margin on the training set can be obtained:

$$\min_{w_j,\, j=1,\ldots,|L|} \sum_{l=1}^{|L|} ||w_l||^2 \tag{24}$$

Subject to the constraint:

$$\langle w_l - w_k, \mathbf{x}_i\rangle + b_l - b_k \geq 1, \; (l,k) \in Y * \bar{Y} \tag{25}$$

In order to get the best possible performance the ranking loss should be minimized while the the margin should be maximized. The ranking loss can be minimized by adding the constraint $\langle w_l - w_k, \mathbf{x}_i\rangle + b_l - b_k \geq 1 - \xi_{ilk}, \; (l,k) \in Y * \bar{Y}$ which leads to the ranking loss on the training set to become:

$$\frac{1}{m}\sum_{i=1}^{m} \frac{1}{|Y_i||\bar{Y_i}|} \sum_{(l,k) \in Y * \bar{Y}} \theta(-1 + \xi_{ilk}) \tag{26}$$

Where θ is the Heaviside function which is approximated as just ξ_{ilk}. In order then to both maximize the margin and minimize the ranking loss equations 24 and 26 can be combined to form an optimization problem:

$$\min_{w_j,\, j=1,\ldots,|L|} \sum_{l=1}^{|L|} ||w_l||^2 + C\sum_{i=1}^{m} \frac{1}{|Y_i||\bar{Y_i}|} \sum_{(l,k) \in Y * \bar{Y}} \xi_{ilk} \tag{27}$$

Subject to the constraints:

$$\begin{aligned}&\langle w_l - w_k, \mathbf{x}_i\rangle + b_l - b_k \geq 1,\ (l,k) \in Y * \bar{Y}\\ &\xi_{ilk} \geq 0\end{aligned} \tag{28}$$

Once this optimization problem has been solved the labels can be properly ranked. After the labels have been ranked the threshold value can be calculated with:

$$\begin{aligned}t(\mathbf{x}_i) = &\arg\min_t |\{l \in Y \text{ such that } f_l(\mathbf{x}_i) \leq t|\\ &+ |\{l \in \bar{Y} \text{ such that } f_l(\mathbf{x}_i) \geq t|\end{aligned} \tag{29}$$

Where $(f_1(\mathbf{x}_i), ..., f_{|L|}(\mathbf{x}_i))$ are given by the ranking system.

2.3.4 Multi-label Radial Basis Function Neural Networks

Multi-Label Radial Basis Function (ML-RBF) neural networks is a type of classification method created by Min-Ling Zhang [28] which has been derived from the original RBF neural network method. The explanation of the ML-RBF method is paraphrased from [28] in the text below. The notation used in previous sections is reused for this section.

When creating a ML-RBF neural network the training data is first divided into different sets $U_l = \{\mathbf{x}_i|(\mathbf{x}_i, Y_i) \in D, l \in Y_i\}$, one for each label. A k-means clustering is then done on each of these sets in order to from $k_l = \alpha * |U_l|$ groups for each set where α is chosen to be a suitable fraction. A k-means clustering works by partitioning the data into k different groups where each observation belongs to the group with the closest mean. The mean of each group is calculated from the group's centroid which is the point that minimizes the variance within the group. This means that k_l centroids are created for each label. These centroids are then assigned as the center vectors $\mathbf{c}_j^l$ for the neurons as can be seen in figure 6, which means that there will be a total of $K = \sum_{l=1}^{L} k_l$ neurons in the hidden layer plus the additional bias term. Each of these neurons including the bias has a weight to every output node. These weights can be structured into a weight matrix $W = [w_{jl}]_{j=0,...K}^{l=1,...,L}$ where the $j = 0$ refers to the weights from the bias neuron. This weight matrix can be found by minimizing the sum of squares error:

$$E = \frac{1}{2}\sum_{i=1}^{m}\sum_{l=1}^{L}\left(y_l(\mathbf{x}_i) - t_l^i\right)^2 \tag{30}$$

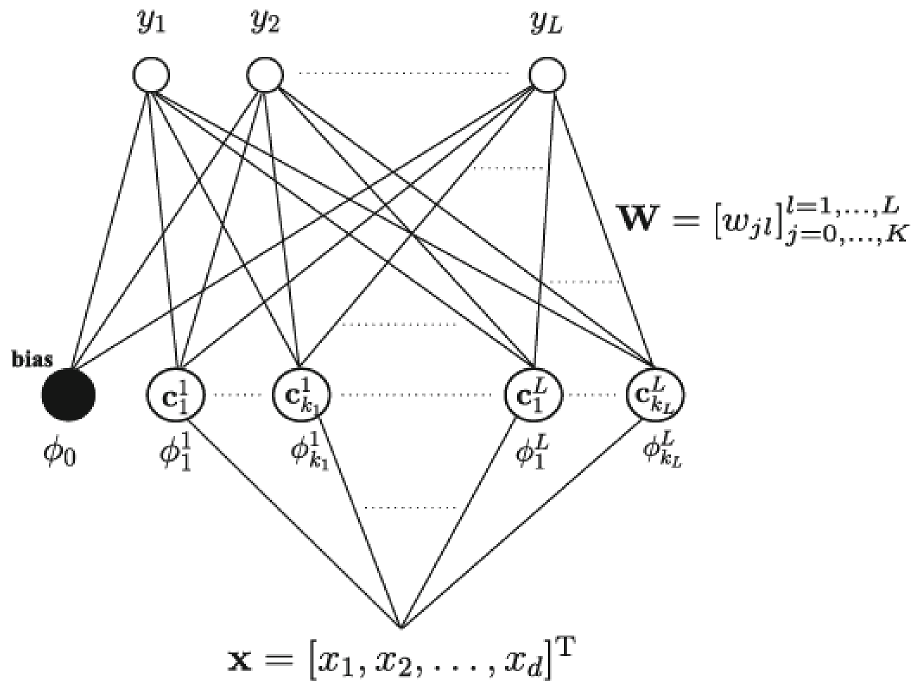

Figure 6: Illustration of the structure of the ML-RBF neural network [28]

Where t_l^i is the known correct output which in contrast to previous sections take the values 1 if $l \in Y_i$ and -1 if $l \notin Y_i$ compared to the 0 used previously to indicate that a label was not in the label set. $y_l(\mathbf{x}_i)$ on the other hand is the predicted output which can also be written as $y_l(\mathbf{x}_i) = \sum_{j=0}^{L} w_{jl}\phi_j^l$ with ϕ_j^l being the activation function of the corresponding neuron as seen in figure 6. For the sake of convenience and to make the notation on later equations easier the indexing of the basis functions and the center vectors are changed. The indexing of the basis functions $\{\phi_0, \phi_1^1, ..., \phi_{k_1}^1, ..., \phi_1^L, ..., \phi_{k_L}^L\}$ is put together into $\phi_j, (0 \leq j \leq K)$ while the center vectors $\{c_1^1, ..., c_{k_1}^1, ..., c_1^L, ..., c_{k_L}^L\}$ is put together into $c_j, (0 \leq j \leq K)$. Before this can be used to calculate the weight matrix the activation functions first have to be defined. In this method the activation functions were chosen to be a Gaussian type function:

$$\phi_j(\mathbf{x}_i) = \exp\Big(- \frac{\mathrm{dist}(\mathbf{x}_i, \mathbf{c}_j)^2}{2\sigma_j^2} \Big) \tag{31}$$

Where $\text{dist}(x_i, c_j)$ measure the euclidean distance between the training data and the centroid. The σ term is a smoothing factor and is equal for all activation functions in this method. The smoothing factor can be calculated using the following equation:

$$\sigma = \mu\Big(\frac{\sum_{p=1}^{K-1}\sum_{q=p+1}^{K}\text{dist}(\mathbf{c}_p, \mathbf{c}_q)}{K(K-1)/2}\Big) \tag{32}$$

With μ being being a suitable chosen scaling factor. Once all of this has been completed the weight matrix can be calculated by differentiating equation 30 with respect to w_{jl} and then setting the derivative to zero. The resulting equation then becomes:

$$(\Phi^T\Phi)W = \Phi^T T \tag{33}$$

With $\Phi = [\phi_{ij}]_{m*(K+1)}$ where $\phi_{ij} = \phi_j(\mathbf{x}_i)$, $W = [w_{jl}]_{(K+1)*L}$ and $T = [t_{il}]_{m*L}$ where $t_{il} = t_l^i$. Once the weight matrix has been calculated the training for the ML-RBF neural network is complete. Predicting the label set of an observation $\mathbf{z}$ can then be done with the following equation:

$$Z = \{l|y_l(z) = \sum_{j=0}^{K} w_{jl}\phi_j(z) > 0, l \in L\} \tag{34}$$

Where Z is the predicted label set.

2.4 Evaluation Metrics

Evaluating the performance of multi-label methods is more complicated compared to single-label methods. Unlike in single-label classification a prediction can be partially correct when using multi-label classification since it is possible for the method to get some labels correct while missing others. There are many different evaluation metrics for multi-label classification and those used in this work are listed below.

Hamming Loss

Hamming Loss gives the ratio of incorrect labels compared to the total amount of labels. This includes both labels that are incorrectly added to the label set and labels that have been incorrectly removed from the label set. The Hamming loss can be calculated with the following equation:

$$HammingLoss = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|L|}|H_i \Delta Y_i| \tag{35}$$

Where Δ stands for the symmetric difference of two sets, H_i is the set of predicted labels obtained from a multi-label classifier and Y_i is the set of correct labels.

One Error

In many multi-label methods each observation is given a ranked list of labels were a higher rank correlates with a higher chance of the label belonging to the the correct label set according to the method used. One Error shows how often the highest ranked label does not belong the correct label set. One Error can be calculated using the following equation:

$$OneError = \frac{1}{N}\sum_{i=1}^{N}[[\arg\max_{y \in Y} f(x_i, y) \notin Y_i]] \tag{36}$$

Where $\arg\max_{y \in Y_i} f(x_i, y)$ gives the highest ranked label for the observation x_i, Y is the set of all labels, and Y_i is the set of correct labels for observation x_i.

Subset Error

Subset error is equivalent to the error rate used in single-label classification. Thus subset error measures the amount of times that the predicted label set does not match up perfectly to the correct label set and then compares this to the total amount of tested instances.

$$SubsetError = \frac{1}{N}\sum_{i=1}^{N}[H_i \neq Y_i] \tag{37}$$

Where H_i is the set of predicted labels and Y_i is the set of correct labels.

3 Method

In this chapter the methods and process that were used to acquire the results are explained. Section 3.1 covers which tools and equipment were used and how a recording session was carried out. In section 3.2 the signal processing is discussed, what types of filters that were used and which parameters that were chosen for the feature extraction. Finally in section 3.3 it is explained how the different classifiers were used and how they were configured. The programming language of choice in this work was Matlab.

3.1 EMG Recording

Before any classification or signal processing could be done the EMG data first had to be gathered. There are two different ways to gather EMG data. The first way is done by placing electrodes on the subjects skin and then recording the activity of the muscles beneath it, this method is called surface EMG. Surface EMG has the advantage that it is easy to perform and does not require any invasive procedures but in return there is an increase in noise and a loss of signal clarity since the signal is measured through the skin. The second way is done by inserting electrodes directly into the muscle tissue, this method is called intramuscular EMG. The major advantage of this method is the increase in signal quality compared to surface EMG but this procedure is also far more invasive. In this work only surface EMG was recorded and used.

Two different types of recording equipment were used: the Myo and the Quattrocento. For both of these devices the same two arm positions were used when obtaining data. In one of the positions the elbow was resting on a table with the arm being in a raised position, this position is referred to as the free position. For the other position the arm was placed in a stand, see figure 8, so that a clearer distinction between the movement of individual fingers could be obtained.

Initially different hand and wrist movements were considered for testing multi-label classification but this was later changed in favor of finger flexion movements. The most commonly tested movement orders in this work were three and four finger movements. Five finger movements were also tested but not to the same extent due to the greatly increased number of finger combinations which corresponded with a higher degree of errors during recording sessions.

The standard structure used for the training set movement orders in this work was five repetitions of each individual finger movement followed by one repetition of each finger combination movement. The standard movement order for testing sets was one repetition for both individual finger movements and finger combination movements. An illustration of the structure for training and testing data can be seen in figure 7. Some testing was also done when the training set only contained individual finger movements without any combinations which was then tried on the standard testing set. The reason for this was to determine if information gained from the individual movements could be used when trying to classify finger combinations. The process of recording

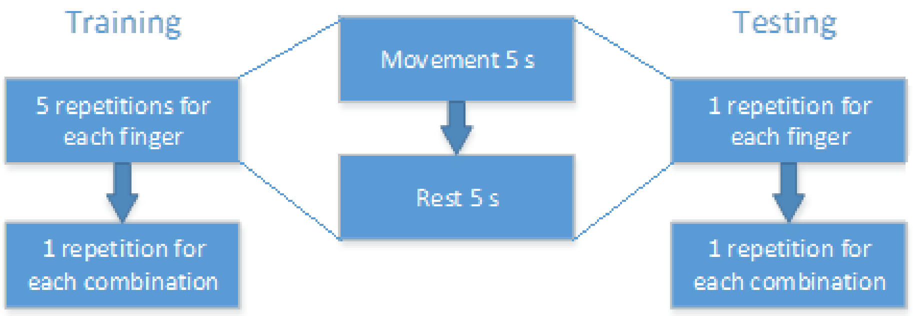
Training
5 repetitions for each finger
1 repetition for each combination
Movement 5 s
Rest 5 s
Testing
1 repetition for each finger
1 repetition for each combination

Figure 7: Flowchart of movements for training and testing data.

a set of EMG data was done by first picking a movement order to follow and then a position for the arm. The subject was then instructed to imitate a sequence of images depicting each movement in the movement order. The subject was instructed to keep putting force into the movement for the entire movement phase. During rest phases the subject was instructed to relax their arm and move as little as possible. When the stand was used the subject was instead instructed to put force towards attempting the movement since the stand prevents the fingers from actually moving. All of these movement orders followed the same pattern of finger movement followed by a return to rest state. Time spent in each movement and in rest state was initially selected to be three seconds but was later changed to five seconds in order to decrease error due to the transition period between movements.

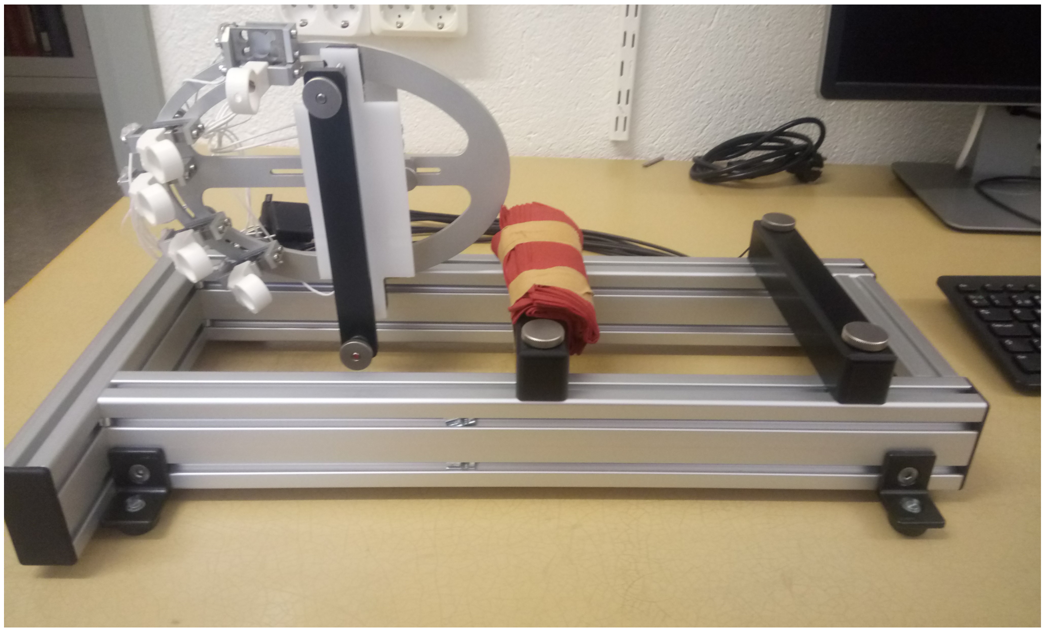
Figure 8: The arm stand

3.1.1 The Myo

The Myo, see figure 9, is a wearable gesture and motion control device developed by ThalmicLabs [14]. While the Myo does have its own classification system and the ability to record IMU data in this work only the EMG recording capabilities are used. The Myo can record EMG signals with a sampling rate of 200 Hz from 8 channels [25].
Recording with the Myo was done by first placing it around the forearm of the subject close to the elbow. The Myo was placed so that the electrodes had direct contact with the skin without any conductive gel in between. The subject was then instructed to perform some hand movements in order to synchronize the Myo with the computer. Once synchronization had been achieved and the Myo had been given some time to warm up the recording session could be started. A Matlab program was then used to pull data from the Myo while recording.

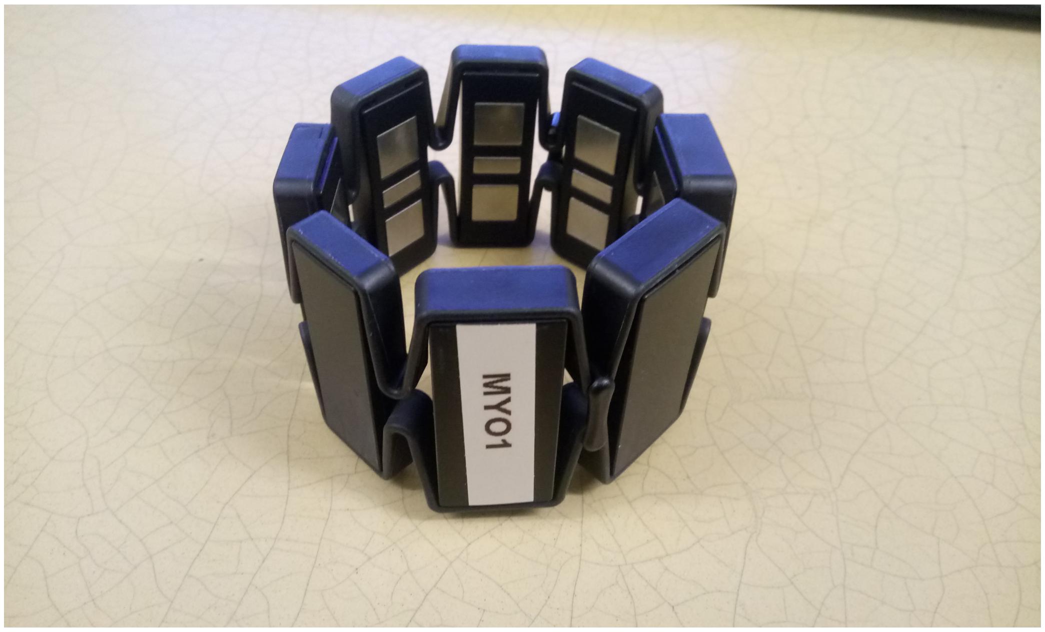

Figure 9: The Myo created by Thalmic Labs.

3.1.2 The Quattrocento

The Quattrocento, see figure 10 is a bio electrical amplifier created by OT Bioelettronica [1] which in this work was used in conjunction with two 8x8 high density surface EMG electrodes to record EMG data. Before any data could be gathered the Quattrocento first had to be configured. The sampling rate of the Quattrocento was chosen to be 2048 Hz while the channel setup was selected to be monopolar with a differential configuration. Originally a Matlab program was used to pull data from the Quattrocento so that it could be quickly processed but this method suffered from a desynchronization between the EMG data and the class data. In order to solve this problem the OT BioLab program was used instead to gather the data with a computer generated trigger signal added to the Quattrocento that indicated when the recording started and when the movement class changed.
Once the configuration had been completed the electrodes were coated with an adhesive and conductive gel. The electrodes were then placed on the subjects arm with one electrode placed close to the elbow where it covered the flexor digitorum superficialis and the flexor digitorum profundus while the other electrode was placed on the wrist where it covered the flexor pollicis

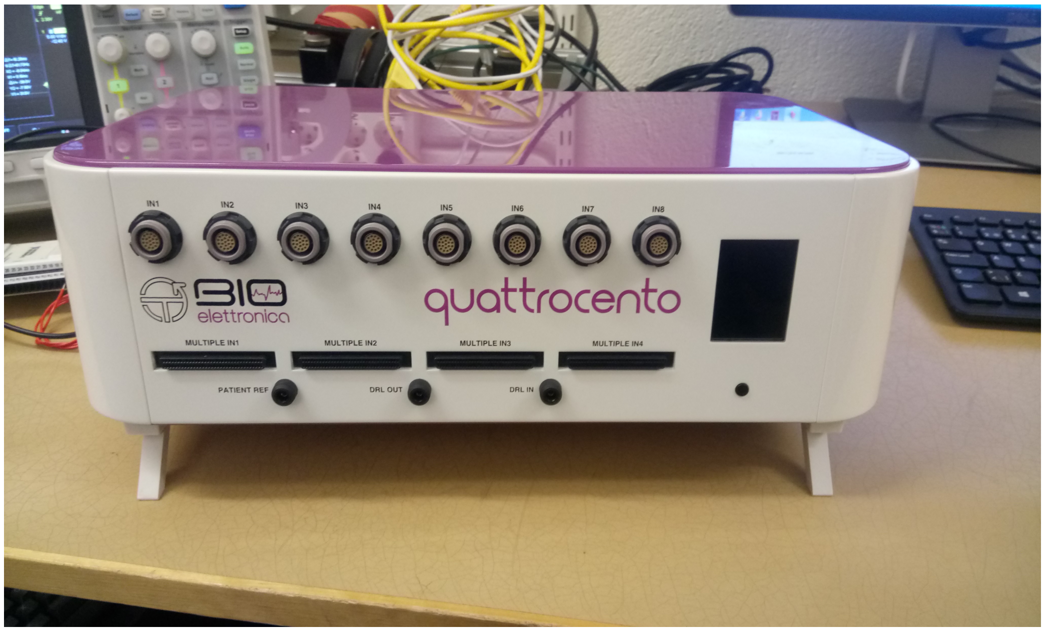

Figure 10: The Quattrocento by OT Bioelettronica

longus. After this was done a grounding bracelet was placed around the subjects wrist and connected to the Quattrocento patient reference port. Finally a NI USB-6009 card used for generating the trigger signals from the computer to the Quattrocento was connected to the auxilliary input. Once these steps had been completed the recording session could be started. An example of the recording setup can be found in figure 13 while an example output from a recording session using the Quattrocento can be seen in figure 11.

3.2 Signal Processing

The first part of the signal processing done on the data was filtering. Both the Myo and the Quattrocento have inbuilt filtering for 50/60 Hz power line interference [24][1]. For the Myo the low frequency noise was handled by a 6th order high pass Butterworth filter with a cutoff frequency of 5 Hz. A low pass filter with a cutoff frequency of 100 Hz was also tested to see if aliasing could have a negative impact on the classification due to the Myo's limited sampling rate. The results showed that the addition of this low pass filter did

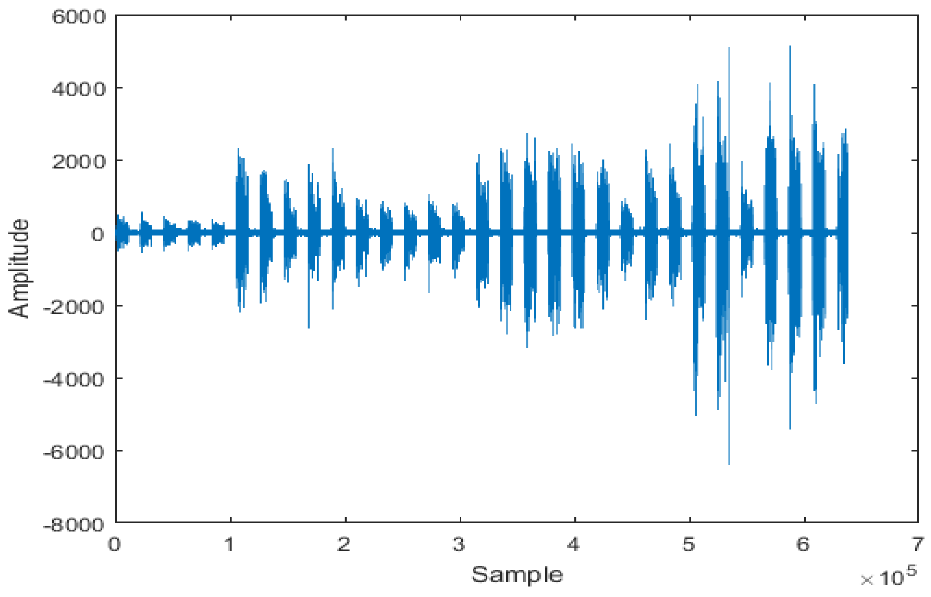

Figure 11: Example of an EMG signal recorded by the Quattrocento

not affect the classification accuracy of any of the used evaluation methods. In the end the decision was made to not include any low pass filter for Myo data. The Quattrocento data on the other hand was filtered by both a low pass and a high pass filter in addition to the inbuilt power line filtering. The high pass filter used for the Quattrocento data was a 6th order Butterworth filter with the cutoff frequency chosen to be 5 Hz. The low pass filter was also a 6th order Butterworth filter with a cutoff frequency of 600 Hz. An example of the frequency content of an EMG signal can be observed in figure 12.

After having filtered the signals feature extraction was performed in order to obtain useful information from the signals which also had the benefit of reducing the amount of data that was later put into the classifier. In the case of the Myo a uniform window with a size of 200 data points (1000ms) and an increment of 20 (100ms) for every step was used for feature extraction. Since the Quattrocento data was sampled with a much higher sampling rate a larger window was used. The window size for the Quattrocento data was 400 data points (approximately 200ms) with an increment of 40 (approximately 20ms) for each step, as with the Myo data a uniform window was used.

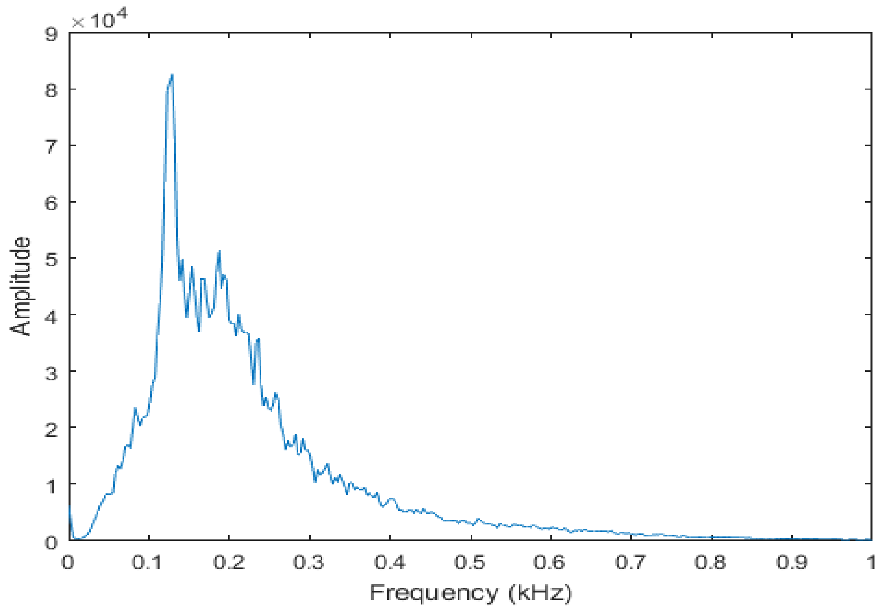

Figure 12: Example of the frequency content of an EMG signal recorded by the Quattrocento.

The extracted features have a great impact on the performance of a classifier. A good set of features can allow a bad classifier to achieve passable classification accuracy while a bad set of features can make even the best classifier perform poorly. In this work 10 features were chosen, see section 2.1. These features were chosen so that they covered different properties of interest in the signal. The choice of how many features to use was based on if any increase in classification performance was noted when a new feature was added. A notable drawback of adding features though is that the amount of predictors that are given as input to the classifier increases which in turn leads to an increase in the training time.
An idea that came up during this process was that combinations of some of the chosen features might achieve better results compared to the combination of all chosen features. If this was the case then it would not only lead to better classification performance but also to a shorter training time. In order to test this idea a feature optimization program was created. This program simply tested all possible feature combinations from one to ten features and then sorted the combinations from highest to lowest accuracy.

3.3 Classification

The single-label methods used in this work were explained in section 2.2 and were implemented by making use of the fitc*-type functions in Matlab. The parameters for these methods were chosen to be the default parameters and the number of neighbours for the KNN algorithm was chosen to be 30 after some testing.

The multi-label methods used in this work were explained in section 2.3 and were implemented by making use of code created by Min-Ling Zhang [27]. For the MLKNN method the number of neighbours used was chosen to be 30 as was done for the single-label KNN method. The other parameters for the MLKNN method were chosen to be the default values presented in [30]. For the ML-RBF method the default values were used as parameters since changes to these values did not produce any noticeable increase in classification accuracy. For the Rank-SVM method a linear kernel was used, the cost for observations placed on the wrong side of the margin was chosen to be 0.1, the maximum number of iterations was chosen to be 30 and the default values provided in the code [27] were used for all other parameters. These parameters were chosen after some testing was done but the results obtained varied little even when radically different parameter values were used. For the MLNB method some additional preprocessing of the data was performed before any classification was done. This additional preprocessing took the form a principal component analysis combined with a genetic algorithm as suggested by [29]. The parameters used for the preprocessing steps and for the MLNB classification method were the default values suggested in the code by Zhang [27].

After the data had been classified some form of evaluation had to be done in order to determine whether or not the results were good. The different types of evaluation methods used in this work were explained in section 2.4. There was one problem that became evident when doing these evaluation which was that the classification of the rest states obfuscated how good the total classification performance actually was. The reason for this was that around half of all observations were rest states which were very easy to classify. This meant that a classification could fail to classify any finger movement correctly and still achieve around 50% accuracy as long as it could correctly classify the rest states. Therefor in order to get a better view of how each method performed the subset error was also calculated on the predicted results with all rest states excluded.

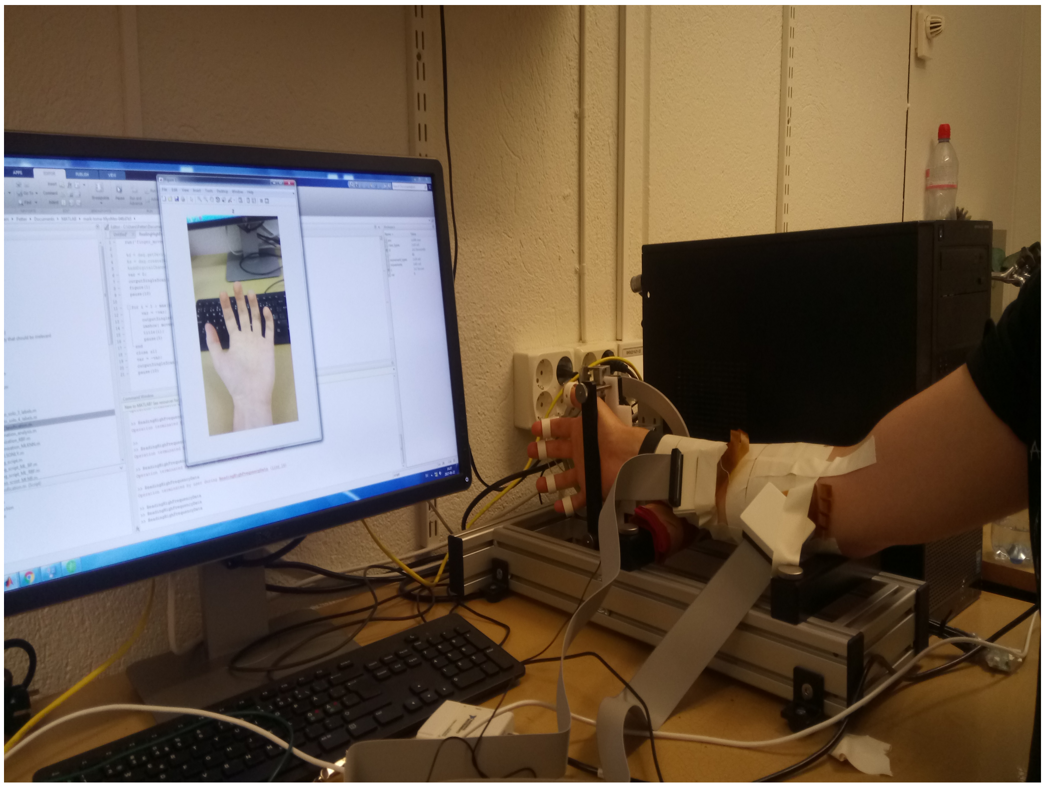
Figure 13: Example of an EMG recording session using the Quattrocento

4 Results

4.1 Feature Optimization

In this section the results obtained from the feature optimization program are presented. Due to the large amount of time required to do a feature optimization only results using Myo data are included. Figure 14 illustrates the frequency of appearance for each feature in the top 100 feature combinations when using the MLKNN classifier. The top 100 feature combinations refer to the 100 combinations of features that obtained the highest subset accuracy. Wavelength for example appeared in around 70 different feature combinations out of the 100 most accurate while frequency median only appeared in around 20. In figure 15 it is shown how common the quantity of features in each feature combination is for the top 100 combinations when using the MLKNN classifier. As an example this means that out of the top 100 most accurate feature combinations 22 feature combinations contained 5 features while none contained 1, 9 or 10 features. Figures 16 and 17 show the same results as figures 14 and 15 but when using the ML-RBF classifier instead. The results shown in figures 14 - 17 were obtained from testing done on the same data set. Results from running the feature optimization program on other data sets can be found in the appendix.

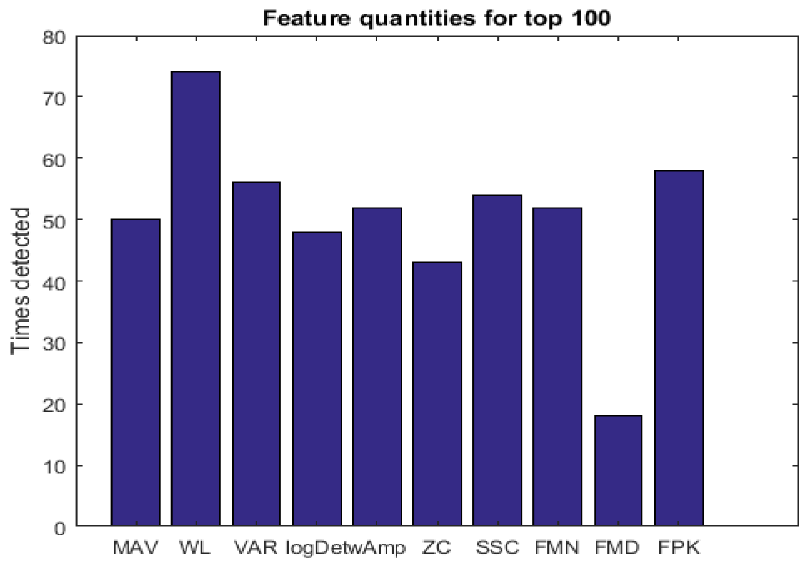

Figure 14: Top 100 feature quantities for MLKNN

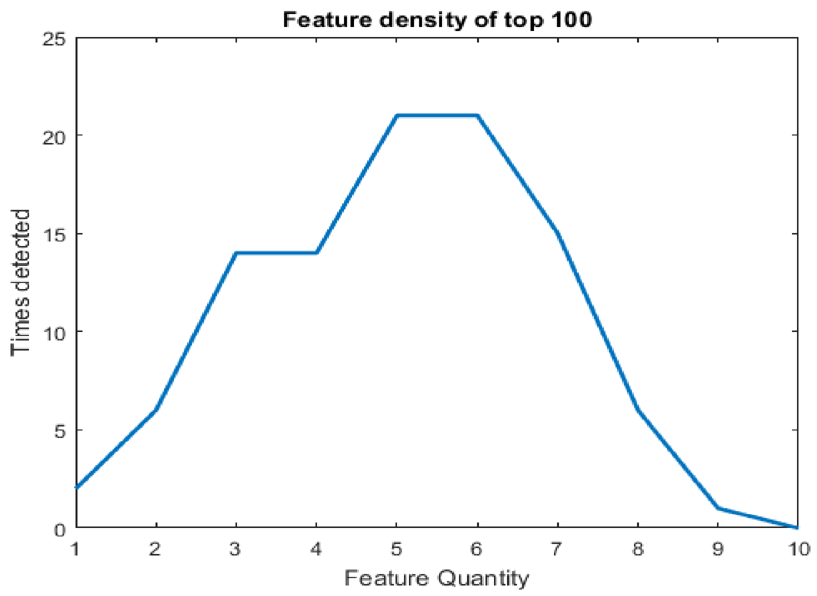

Figure 15: Top 100 feature densities for MLKNN

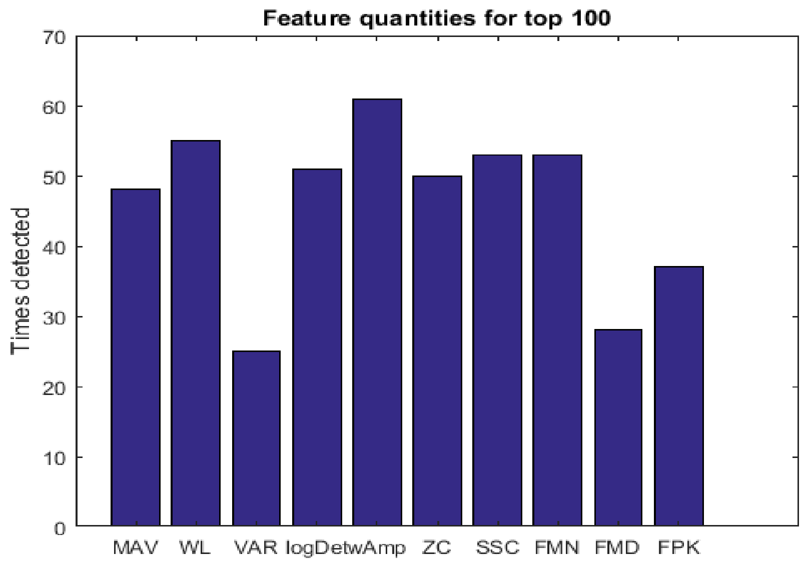

Figure 16: Top 100 feature quantities for ML-RBF

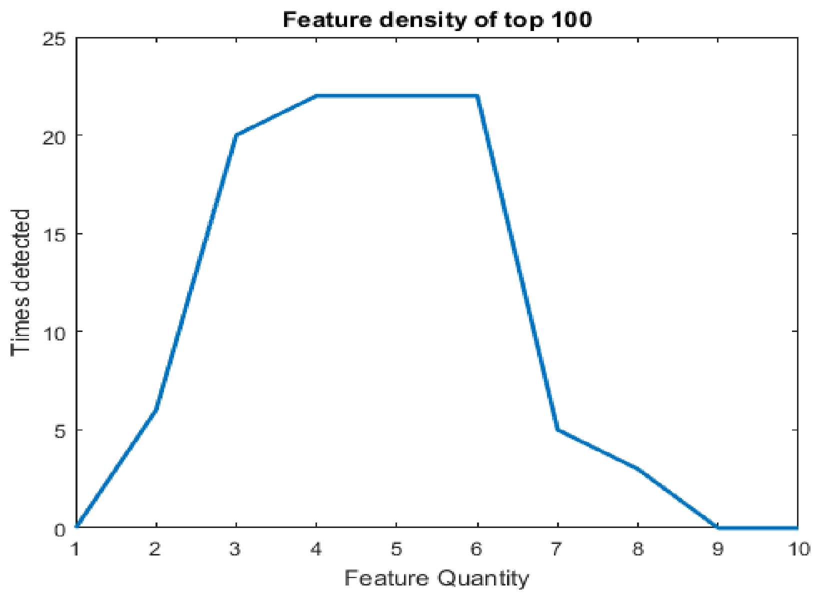

Figure 17: Top 100 feature densities for ML-RBF

4.2 Single-label Results

In this section the results obtained from testing the different single-label classification methods are presented. The data sets used in this section follow the standard movement order discussed in section 3.1. All 10 features were used when obtaining the results in this section. Tables 2 - 4 present the classification accuracy of each method depending on which arm position and measuring device that was used. Figures 18 - 21 show examples of the results obtained from each of the classification methods when using data obtained from the Quattrocento and with the arm placed in the free position. In these four figures the blue lines show the results that the classification methods have predicted while the red lines show what the correct class is. The places where the red and blue line do not coincide are thus samples were the classifiers have made incorrect predictions. The class # refers to which movement was made with 1 = Thumb (T), 2 = Index (I), 3 = Long (L), and 4 = Ring (R). The other class numbers are explained in table 1

Class	5	6	7	8	9	10	11	12	13	14	15
Movement	TI	TL	TR	IL	IR	LR	TIL	TIR	TLR	ILR	TILR

Table 1: Class number and corresponding movement

Classification method	Error	Error (excluding rest states)
KNN	0.3147	0.4964
LDA	0.3481	0.5835
SVM	0.4106	0.6513
NB	0.3649	0.5654

Table 2: Results from single-label classification methods on 4 finger Myo data with arm placed in stand.

Classification method	Error	Error (excluding rest states)
KNN	0.2878	0.4875
LDA	0.1783	0.2835
SVM	0.9740	0.9498
NB	0.2732	0.4411

Table 3: Results from single-label classification methods on 4 finger Quattrocento data with arm placed in stand.

Classification method	Error	Error (excluding rest states)
KNN	0.2535	0.3902
LDA	0.1562	0.2126
SVM	0.6685	0.8632
NB	0.3253	0.4678

Table 4: Results from single-label classification methods on 4 finger Quattrocento data with arm in free position.

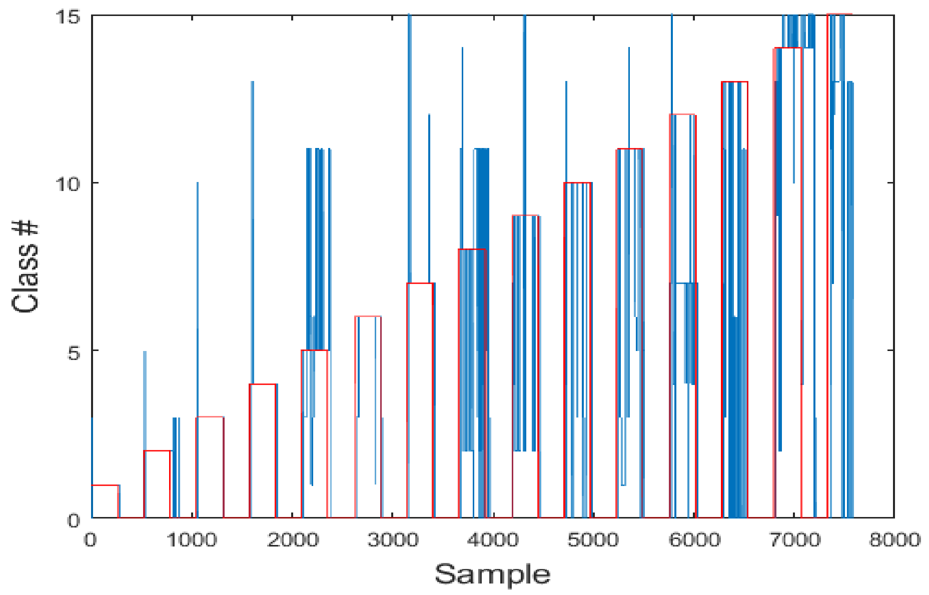

Figure 18: Results from a KNN classification on Quattrocento data with with arm in free position. 4 fingers were used in this data. The blue line shows the predicted results while the red line shows the ground truth

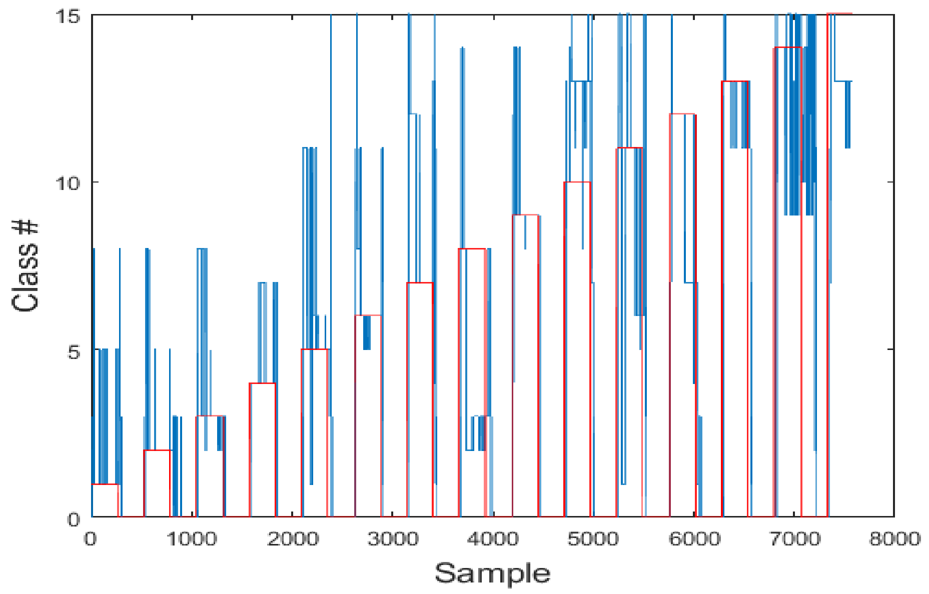

Figure 19: Results from a NB classification on Quattrocento data with arm in free position. 4 fingers were used in this data. The blue line shows the predicted results while the red line shows the ground truth

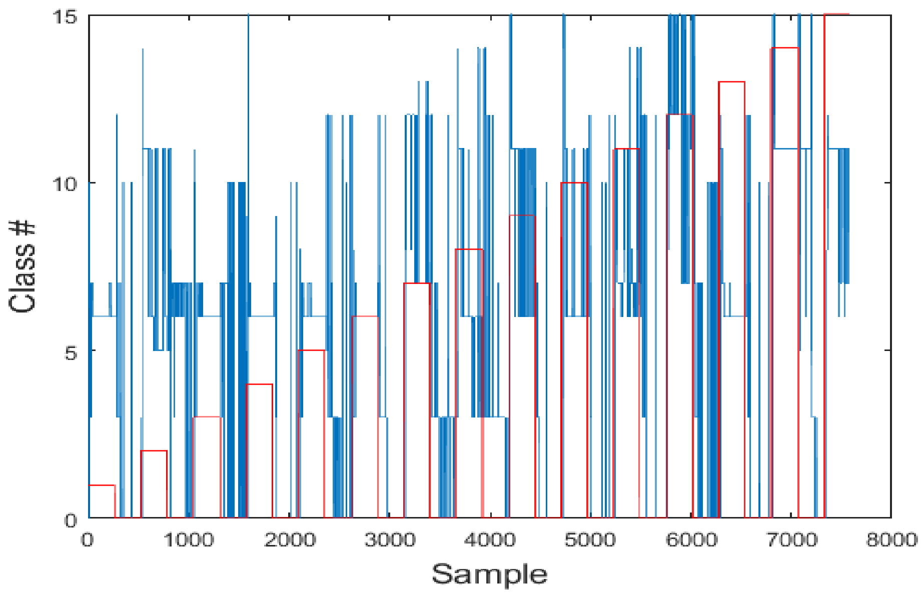

Figure 20: Results from a SVM classification on Quattrocento data with arm in free position. 4 fingers were used in this data. The blue line shows the predicted results while the red line shows the ground truth

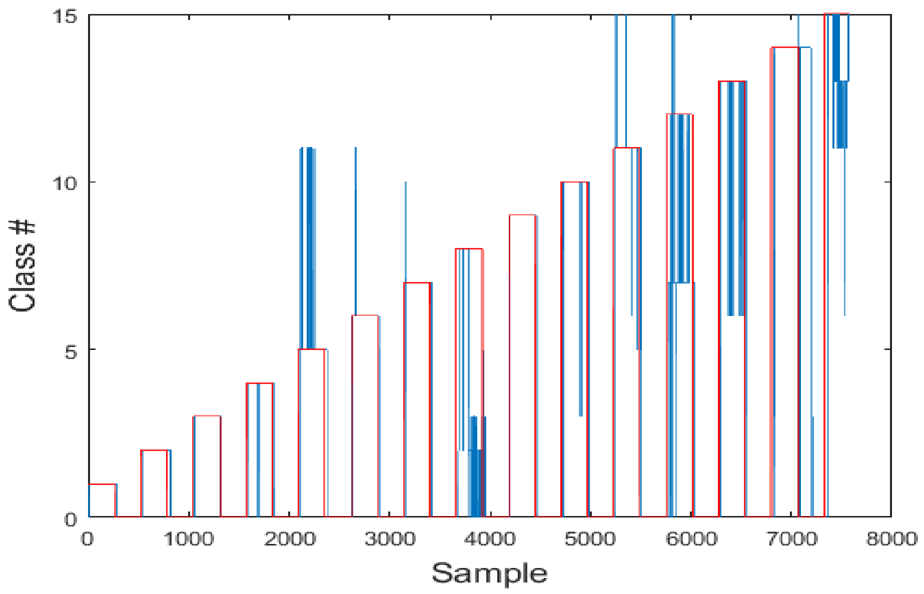

Figure 21: Results from a LDA classification on Quattrocento data with arm in free position. 4 fingers were used in this data. The blue line shows the predicted results while the red line shows the ground truth

4.3 Multi-label Results

In this section the results obtained from testing the different multi-label classification methods are presented. The data sets used in this section follow the standard movement order mentioned in section 3.1. All 10 features were used when obtaining the results in this section. Tables 5-7 give the classification accuracy results from the tested multi-label methods depending on arm position and recording device. Figure 22 shows the results from a MLKNN classification when using data gathered by the Myo and with the arm placed in the stand. Figure 23 gives the results from a ML-RBF classification on a data set recorded by the Quattrocento with the arm held in the free position. Plots of the other multi-label methods can be found in the appendix.

Classification method	Hamming Loss	One Error	Subset Error	Subset Error (excluding rest states)
MLKNN	0.1417	0.0502	0.3385	0.5127
ML-RBF	0.1310	0.1145	0.3288	0.5175
Rank-SVM	0.5120	0.1602	0.9288	0.8951
MLNB	0.1955	0.1004	0.4183	0.7250

Table 5: Results from multi-label classification methods on 4 finger Myo data with arm placed in stand.

Classification method	Hamming Loss	One Error	Subset Error	Subset Error (excluding rest states)
MLKNN	0.1009	0.0488	0.2796	0.4531
ML-RBF	0.0704	0.0447	0.1892	0.3027
Rank-SVM	0.5510	0.2546	0.9306	0.8695
MLNB	0.1772	0.0502	0.4551	0.7274

Table 6: Results from multi-label classification methods on 4 finger Quattrocento data with arm placed in stand.

Classification method	Hamming Loss	One Error	Subset Error	Subset Error (excluding rest states)
MLKNN	0.1112	0.0456	0.2760	0.3907
ML-RBF	0.0777	0.0330	0.1807	0.2552
Rank-SVM	0.4310	0.2203	0.9402	0.8882
MLNB	0.1849	0.0773	0.4300	0.7331

Table 7: Results from multi-label classification methods on 4 finger Quattrocento data with arm placed in free position

4 fingers were used in this data: thumb (T)(blue), index (I)(red), long (L)(yellow) and ring (R)(purple).

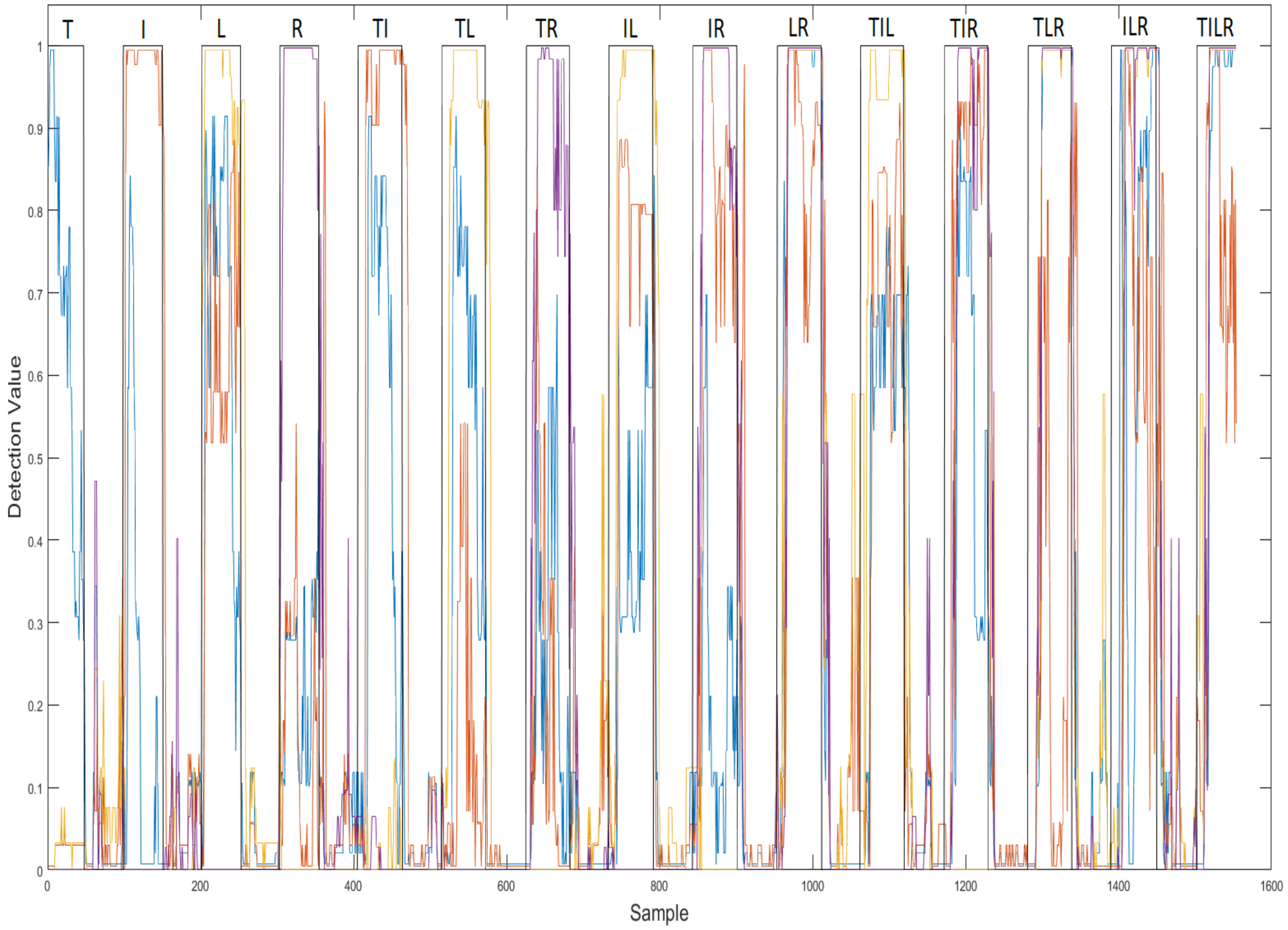

Figure 22: Results from a MLKNN neural network classification on Myo data with the arm placed in the stand. The black line together with the letters shows the correct label set. Values above 0.5 indicate a detected label according to the method.

4 fingers were used in this data: thumb (T)(blue), index (I)(red), long (L)(yellow) and ring (R)(purple).

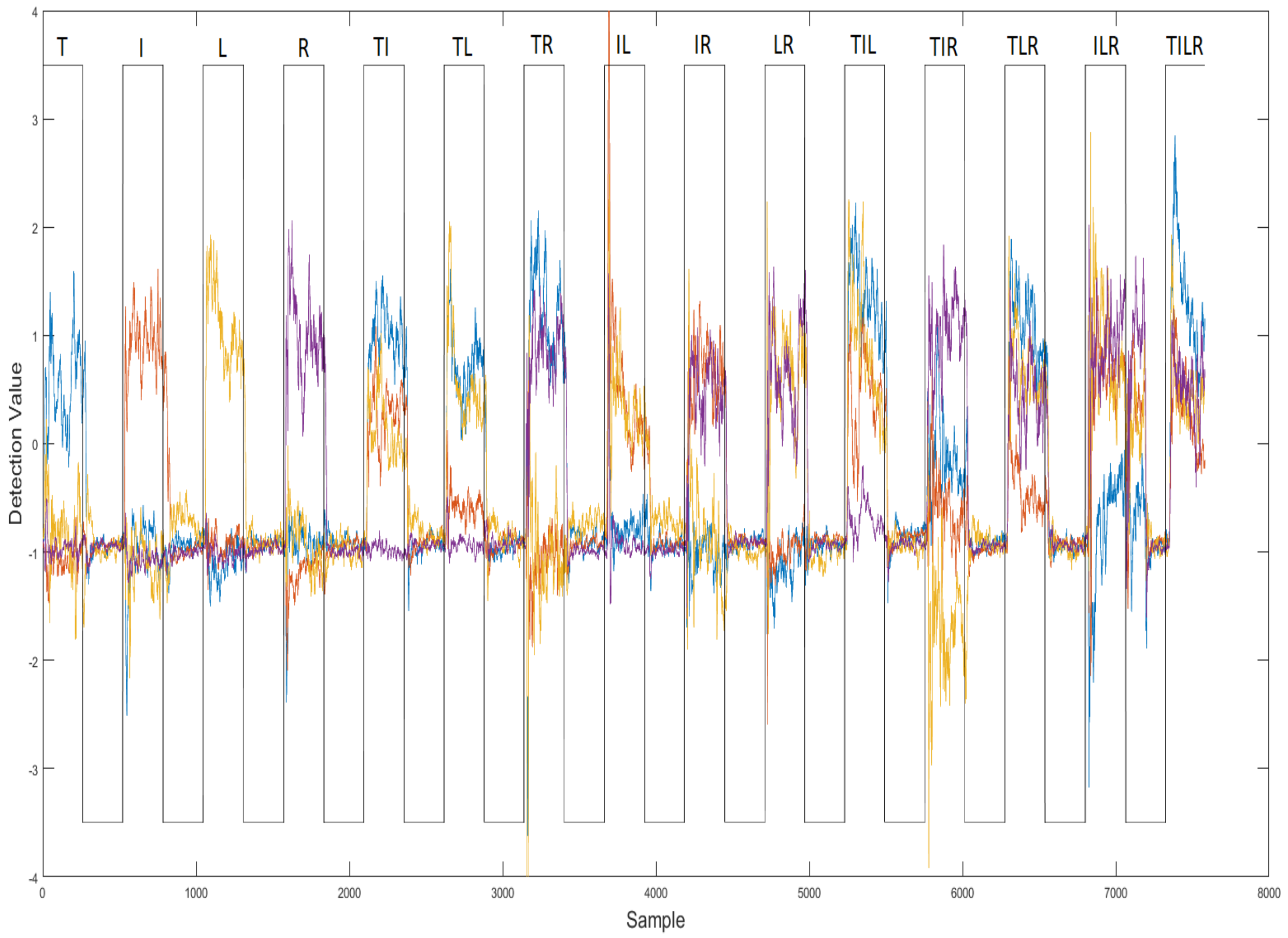

Figure 23: Results from a ML-RBF neural network classification on Quattrocento data with the arm in free postion. The black line together with the letters shows the correct label set. Values above 0 indicate a detected label according to the method.

4.4 Individual label training

In this section the results from using only individual labels as training data and only label combinations as testing data are presented. The movement order used for these tests differs from the previous sections. The training set used in this section consisted of 5 repetitions of each individual finger movement and no combination movements. The testing set contained one repetition of each combination movement and no individual finger movements. All 10 features were used when obtaining the results in this section. Tables 9 - 10 give the classification accuracy results from the tested multi-label methods depending on arm position and recording device. Table 11 is included as an example to show how single-label methods have problems handling this type of classification. Figures 24 and 25 show some results for how the ML-RBF method can handle this type of classification.

Classification method	Hamming Loss	One Error	Subset Error	Subset Error (excluding rest states)
MLKNN	0.2817	0.1183	0.5886	1.0000
ML-RBF	0.2930	0.2209	0.5974	0.9825
Rank-SVM	0.3864	0.2657	0.9868	0.9613
MLNB	0.2862	0.2104	0.5825	0.9968

Table 8: Results from using multi-label classification on 4 finger Myo data with arm placed in stand using only individual finger movements as training data and then testing on finger combination movements.

Classification method	Hamming Loss	One Error	Subset Error	Subset Error (excluding rest states)
MLKNN	0.2254	0.0298	0.5644	0.9930
ML-RBF	0.1971	0.0298	0.4838	0.8593
Rank-SVM	0.5110	0.2881	0.9909	0.9688
MLNB	0.2904	0.2946	0.5794	0.9983

Table 9: Results from using multi-label classification on 4 finger Quattrocento data with arm placed in stand using only individual finger movements as training data and then testing on finger combination movements.

Classification method	Hamming Loss	One Error	Subset Error	Subset Error (excluding rest states)
MLKNN	0.2550	0.0620	0.5941	0.9993
ML-RBF	0.2060	0.0383	0.4580	0.7959
Rank-SVM	0.4964	0.0937	0.9695	0.9329
MLNB	0.2872	0.1086	0.6357	0.9711

Table 10: Results from using multi-label classification on 4 finger Quattrocento data with arm in free position using only individual finger movements as training data and then testing on finger combination movements.

Classification method	Error	Error (excluding rest states)
KNN	0.5828	1.0000
LDA	0.5806	1.0000
SVM	0.5907	1.0000
NB	0.6178	1.0000

Table 11: Results from single-label classification methods on 4 finger Quattrocento data with arm in free position using only individual finger movements as training data and then testing on finger combination movements.

4 fingers were used in this data: thumb (T)(blue), index (I)(red), long (L)(yellow) and ring (R)(purple).

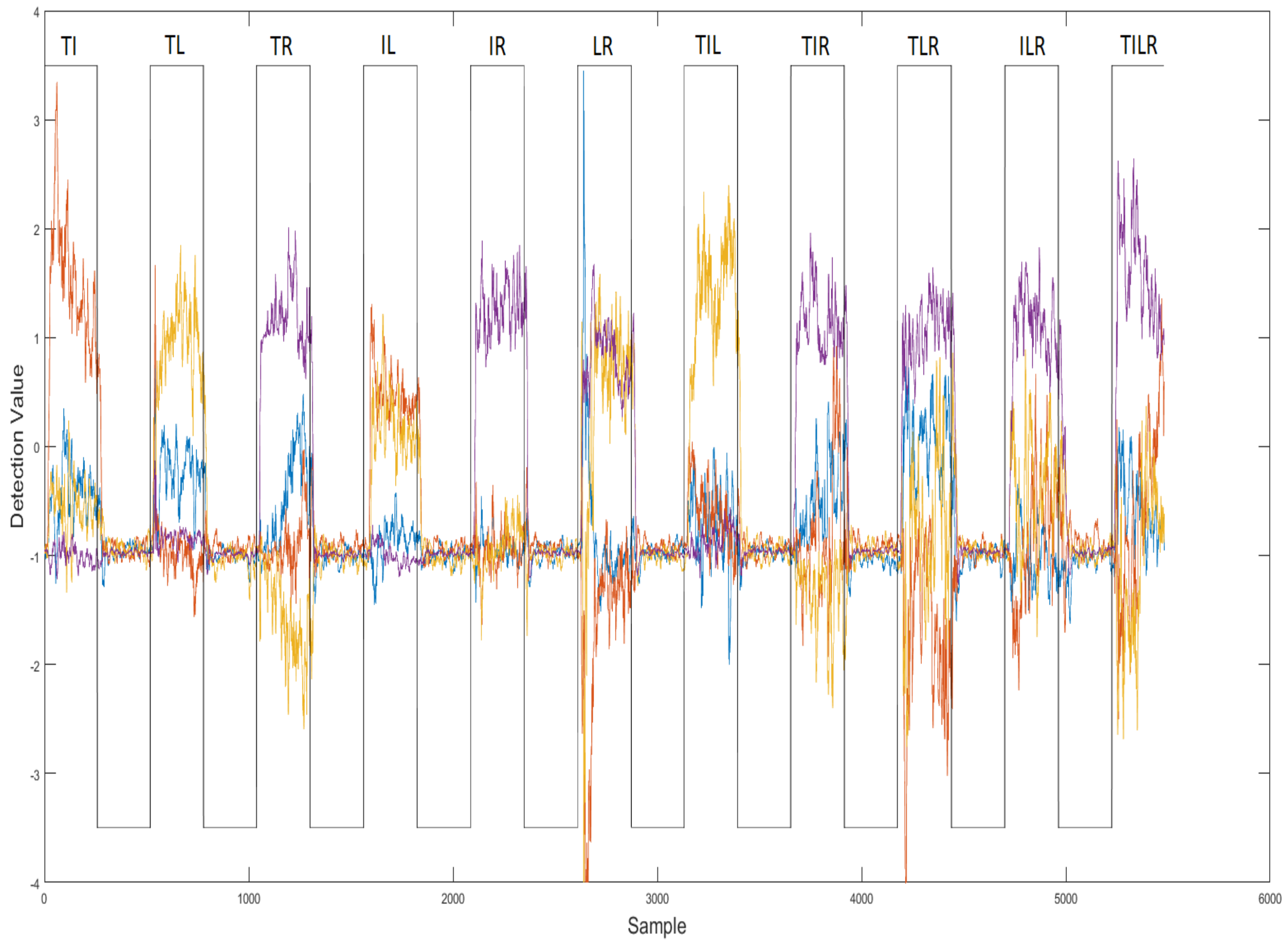

Figure 24: Results from ML-RBF classification on Quattrocento data with the arm placed in the stand using only the individual finger movements as training data. The black line together with the letters shows the correct label set. Values above 0 indicate a detected label according to the method.

4 fingers were used in this data: thumb (T)(blue), index (I)(red), long (L)(yellow) and ring (R)(purple).

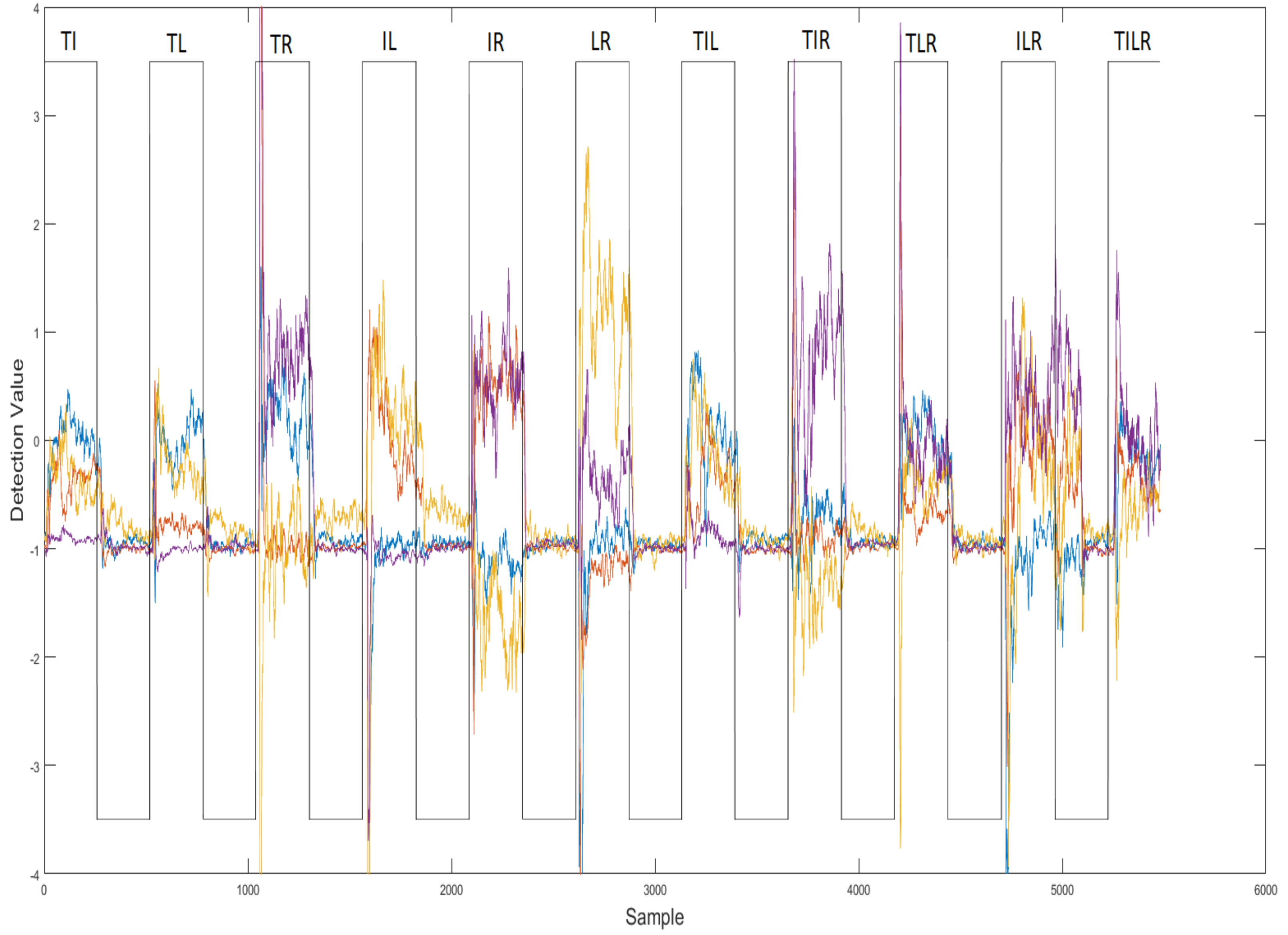

Figure 25: Results from ML-RBF classification on Quattrocento data with arm in free positon using only the individual finger movements as training data. The black line together with the letters shows the correct label set. Values above 0 indicate a detected label according to the method.

5 Discussion

The aim of this work was to investigate the use of multi-label classification methods for the purpose of classifying electromyographic signals. The results from the multi-label methods can be seen in tables 5 - 7 while those from the single-label methods can be found in tables 2 - 4. It can be seen from these tables that multi-label methods do not have any inherently better classification accuracy than single-label methods. Comparing the subset errors of the multi-label methods for both Myo and Quattrocento data to the error rates of the single-label methods shows that both types have similar performance for their best methods. In the values presented the best single-label methods are slightly better then the best multi-label methods but this tended to vary depending on which data set was used. One thing of note was that the results obtained when using the stand were not better than when the arm was in the free position. This came as a surprise since it was initially assumed that using the stand would make it easier for the subject to focus on the different movements. On the other hand this indicates that multi-label methods like single-label methods are not restricted to any specific arm position in order to work. Another point of interest comes from comparing the Quattrocento and Myo results as it shows that even with increased frequency range and amount of data multi- and single-label methods still have similar error rates for their best methods.

Looking at the results from the individual methods also show some things of interest. As can be seen when looking at tables 2 - 7 the KNN and MLKNN methods give very similar results regardless of what arm position was used or which sampling rate was chosen. This contrasts with the NB and MLNB methods since NB works relatively well for a single-labeling method while its multi-label counterpart has a poor performance even with the additional preprocessing that was used in this work. A similar relationship between solo- and multi-label methods can be seen between SVM and Rank-SVM when looking at the Myo data while for the Quattrocento data both methods perform equally poorly. These results show that some of the single-label methods that work for classifying EMG signals can be converted to multi-label methods without a loss of accuracy. It can also be seen that not all methods can make this transition so a method that works well for single-label classification might not even work at all when it comes to multi-label classification.

In section 4.4 only individual labels were used as training data while the testing set consisted only of label combinations. This was done to see if multi-label methods can extrapolate from knowing only the labels by themselves to understanding combinations of labels. As can be seen from tables 8 - 10 most of the methods do not seem to have this ability but what is very interesting is that ML-RBF actually manages to classify some of the label combinations. It can be seen in figure 24 that ML-RBF succeeds in classifying the index + long finger movement and the long + ring finger movement. In figure 25 ML-RBF manages to classify the thumb + ring finger, the index + long finger, the long + ring finger, and it makes a close attempt for the thumb + long finger, the long + ring finger, the thumb + index + long finger, and the index + long + ring finger movement. While ML-RBF only succeeded in classifying some of these movement combinations and mostly during short intervals within these movements it is still far better than the complete failure of the other methods.

As can be seen i figures 15 and 17 just adding features do not guarantee an optimal classification performance. The results show that for the top 100 feature combinations for both ML-RBF and MLKNN the most represented densities lie in the 3-6 features area. Testing on other data sets also showed similar results. This result is in part due to the fact that only 10 features were used which means that the amount of 5 feature combinations are much greater then the amount of 9 feature combinations. Even then it is still worth noting that for all the tested data sets the 10 feature combination was never seen in the top 100. While these results show that a smaller amount of feature can give better classification compared to a larger amount of features a problem quickly occurred when trying to implement this approach namely which features to keep. It was initially believed that figures 14 and 16 could be used to chose which features to discard by just choosing those that appeared the least amount of times. The problem with this approach was that the quantities of the features in the top 100 seemed to depend entirely on the data set used. One feature that was dominant in one data set could be almost nonexistent in another. Another problem with this approach was that while some feature combinations with fewer features were better then the 10 feature combination some were far worse. In the end this lead to the decision to keep using the 10 feature combination that was initially used.

Each of the two recording devices used in this work came with their own set off advantages and disadvantages. The major advantage of using the Myo was the low amount of preparation needed to record EMG data which allowed for many measurements at different time periods. While the Myo was useful it suffers from some severe limitations. One of the most notable is that the sampling rate of the Myo is locked to 200Hz which can have an impact on the classification accuracy since EMG signals contain useful data in the 200-500Hz region [11]. Another limitation is that the electrodes used in the Myo have large surface ares which means that data from each channel is averaged over many nerves which can lead to information being lost. The Quattrocento does not suffer from these problems since it has good selection of sampling rates and the ability to accommodate many different types of electrodes. In return the setup for the Quattrocento is time consuming and requires the use of conducting gel in order to achieve the best possible performance.

6 Conclusion

In this study the use of multi-label classification methods was tested for the purpose classifying EMG signals. The results obtained showed that some multi-label methods could indeed be used for this task. Unfortunately even the best of the tested multi-label methods could only achieve a classification accuracy comparable to that of single-label methods. Thus there does not seem to be any inherent advantage to multi-label methods if only the classification accuracy is considered. Of note though is that there are indications in the data that some multi-label methods might be able to learn combinations of finger movements while only using individual finger movements as training data. Which is an ability that single-label methods do possess.

There are many things that could be done to expand on what has been tested in this work. For example it would be a good idea to test these methods on more subjects to see if similar results are achieved. Another thing of interest could be to see if the extrapolation of training on individual labels to determining label combinations can be improved if more training data is used. More testing on other neural network methods could be done to see if similar or better results could be achieved compared to that of ML-RBF. Testing could also be done on other non neural network methods to see if any other type of method could achieve a measure of extrapolation.

References

[1] OT Bioelecttronica. *Quattrocento product details*. Visited 2017-09-18. URL: `http://www.otbioelettronica.it/index.php?option=com_content&view=article&id=67&lang=en`.

[2] B. E. Boser, I. M. Guyon, and V. N Vapnik. “A training algorithm for optimal margin classifiers”. In: *COLT92 5th Annual Workshop on Computational Learning Theory* (1992), pp. 144–152.

[3] Matthew R. Boutell et al. “Learning multi-label scene classification”. In: *Pattern Recognition* 37 (9 2004), pp. 1757–1771.

[4] Rubana H. Chowdhury et al. “Surface Electromyography Signal Processing and Classification Techniques”. In: *Sensors* 13 (2013), pp. 12431–12466.

[5] Katharina Demet et al. “Health related quality of life and related factors in 539 persons with amputation of upper and lower limb”. In: *Disability and Rehabilitation* 25.9 (2003), pp. 480–486.

[6] Li Deng and Xiao Li. “Machine Learning Paradigms for Speech Recognition: An Overview”. In: *IEEE Transactions on Audio, Speech, and Language Processing* 21.5 (2013), pp. 1060–1089.

[7] Biddiss E and Chau T. “Upper-limb prosthetics: critical factors in device abandonment”. In: *Am J Phys Med Rehabil.* 86 (2007), pp. 977–987.

[8] André Elisseeff and Jason Weston. “A kernel method for multi-labelled classification”. In: *NIPS’01 Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic* (2002), pp. 681–687.

[9] André Elisseeff and Jason Weston. “Kernel methods for multi-labelled classfication and categorical regression problems”. In: *Technical Report, BIOwulf Technologies* (2001).

[10] T.R. Farrell and R.F. Weir. “A comparison of the effects of electrode implantation and targeting on pattern classification accuracy for prosthesis control”. In: *IEEE Trans. Biomed. Eng.* 55.9 (2008), pp. 2198–2211.

[11] Maria Hakonen, Harri Piitulainen, and Arto Visala. “Current state of digital signal processing in myoelectric interfaces and related applications”. In: *Biomedical Signal Processing and Control* 18 (2015), pp. 334–359.

[12] Amir E. Khandani, Adlar J. Kim, and Andrew Lo. “Consumer credit-risk models via machine-learning algorithms”. In: *Journal of Banking and Finance* 34.11 (2010), pp. 2767–2787.

[13] S. B. Kotsiantis, I. D. Zaharakis, and P. E. Pintelas. “Machine learning: a review of classification and combining techniques”. In: *Artificial Intelligence Review* 26 (2006), pp. 159–190.

[14] Thalmic Labs. *Myo main webpage*. Visited 2017-11-16. URL: `https://www.myo.com/`.

[15] Xiao Luo and A. Nur Zincir-Heywood. “Evaluation of Two Systems on Multi-class Multi-label Document Classification”. In: *Foundations of Intelligent Systems. ISMIS* (2005).

[16] Zardoshti-Kermani M et al. “EMG feature evaulation for movement control of upper extrmity prostheses”. In: *IEEE Trans Rehabil Eng* 3 (1995), pp. 324–333.

[17] Silvestro Micera, Jacopo Carpaneto, and Stanisa Raspopovic. “Control of Hand Prostheses Using Peripheral Information”. In: *IEEE Reviews in Biomedical Engineering* 3 (2010), pp. 48–68.

[18] M.Kanevski et al. “Environmental data mining and modeling based on machine learning algorithms and geostatistics”. In: *Environmental Modelling and Software* 19.9 (2004), pp. 845–855.

[19] Science museum. *Iron artifical arm, 1560-1600*. Visited 2017-11-22. URL: `https://collection.sciencemuseum.org.uk/objects/co126844/iron-artificial-arm-europe-1560-1600-artificial-arm`.

[20] Fernando E. B. Otero, Alex A. Freitas, and Colin G. Johnson. “A hierarchical multi-label classification ant colony algorithm for protein function prediction”. In: *Memetic Computing* 2 (3 2010), pp. 165–181.

[21] Ottobock. *Michelangelo prosthetic hand*. Visited 2017-11-22. URL: `https://www.ottobockus.com/prosthetics/upper-limb-prosthetics/solution-overview/michelangelo-prosthetic-hand/`.

[22] Angkoon Phinyomark, Chusak Limsakul, and Pornchai Phukpattaranont. “A Novel Feature Extraction for Robust EMG Pattern Recognition”. In: *Journal of Computing* 1.1 (2009).

[23] Angkoon Phinyomark et al. “EMG Feature Evaluation for Improving Myoelectric Pattern Recognition Robustness”. In: *Expert Systems with Applications* 40.12 (2013), pp. 4832–4840.

[24] ThalmicLabs. *#MyoCraft: EMG in the Bluetooth Protocol*. Visited 2017-08-30. URL: `http://developerblog.myo.com/myocraft-emg-in-the-bluetooth-protocol/`.

[25] ThalmicLabs. *#MyoCraft: Logging IMU and Raw EMG Data*. Visited 2017-08-27. URL: `http://developerblog.myo.com/myocraft-logging-imu-and-raw-emg-data/`.

[26] Grigorios Tsoumakas and Ioannis Katakis. “Multi-Label Classification: An Overview”. In: *International Journal of Data Warehouse and Mining* 3 (3 2007), pp. 1–13.

[27] Min-Ling Zhang. *Codes for multi-instance learning*. Visited 2017-10-03. URL: `cse.seu.edu.cn/people/zhangml/Resources.htm`.

[28] Min-Ling Zhang. “ML-RBF: RBF Neural Networks for Multi-Label Learning”. In: *Neural Processing Letters* 29 (2009), pp. 61–74.

[29] Min-Ling Zhang, José M.Peña, and Victor Robles. “Feature selection for multi-label naive Bayes classification”. In: *Information Sciences* 179.19 (2009), pp. 3218–3229.

[30] Min-Ling Zhang and Zhi-Hua Zhou. “ML-KNN: A lazy learning approach to multi-label learning”. In: *Pattern Recognition* 40 (2007), pp. 2038–2048.

Appendix

A1 Additional feature optimization results

In this section some additional feature optimization figures are presented to further illustrate how volatile some feature can be. Figures 26-27 belong to one data set while figures 28-29 belong to another.

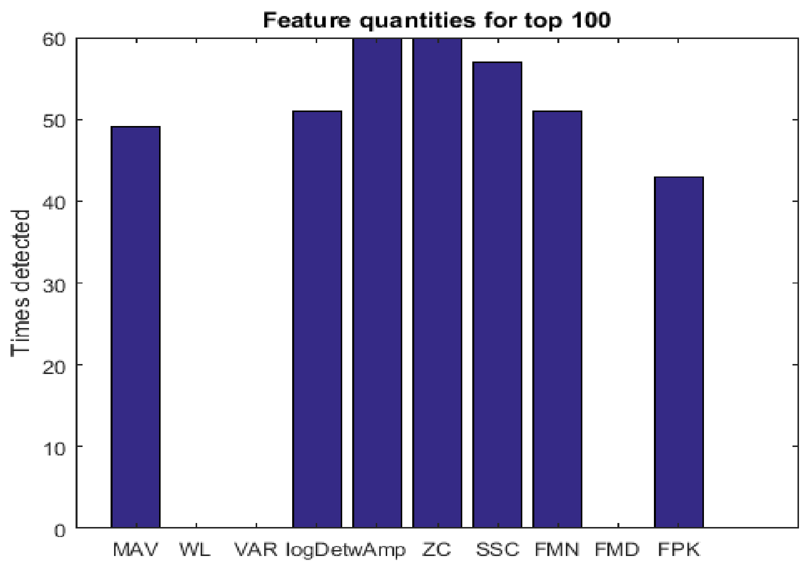

Figure 26: Top 100 feature quantities for MLKNN, data set 1

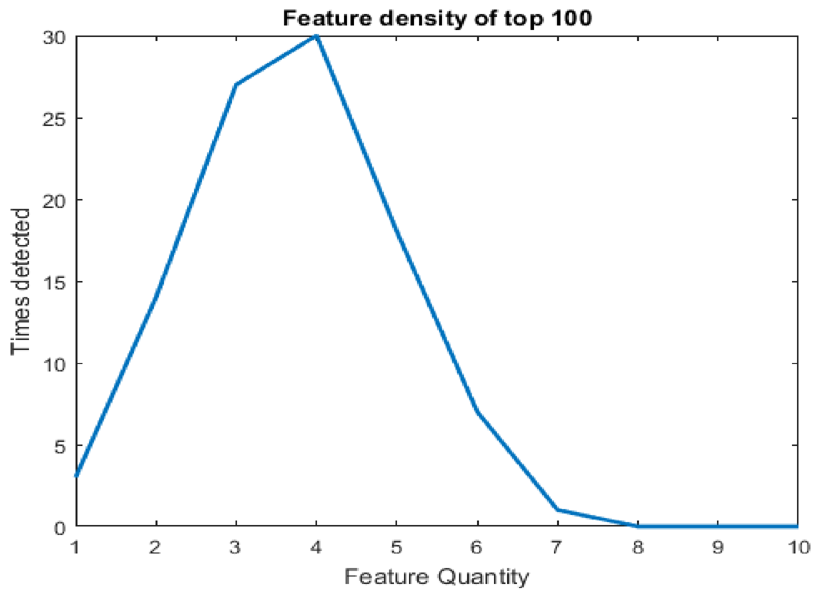

Figure 27: Top 100 feature densities for MLKNN, data set 1

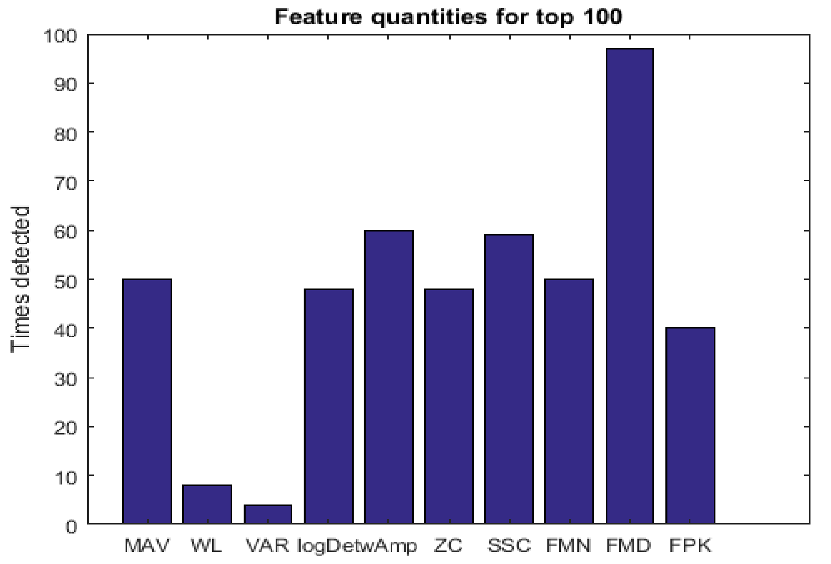

Figure 28: Top 100 feature quantities for MLKNN, data set 2

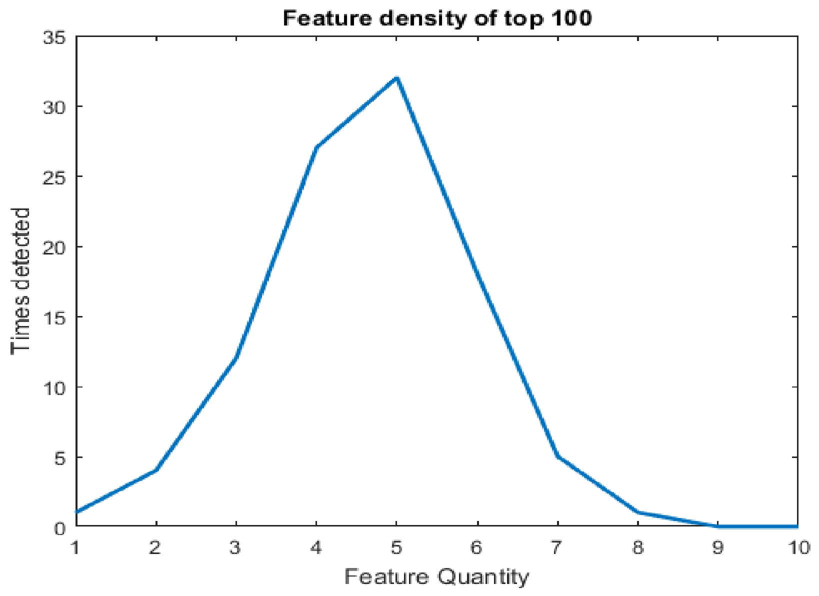

Figure 29: Top 100 feature densities for MLKNN, data set 2

A2 Additional multi-label results

In this section some additional figures of multi-label methods are presented. Figure 30 shows an example of the output from a MLNB classifier while figure 31 shows the output from a Rank-SVM classifier.

4 fingers were used in this data: thumb (T)(blue), index (I)(red), long (L)(yellow) and ring (R)(purple).

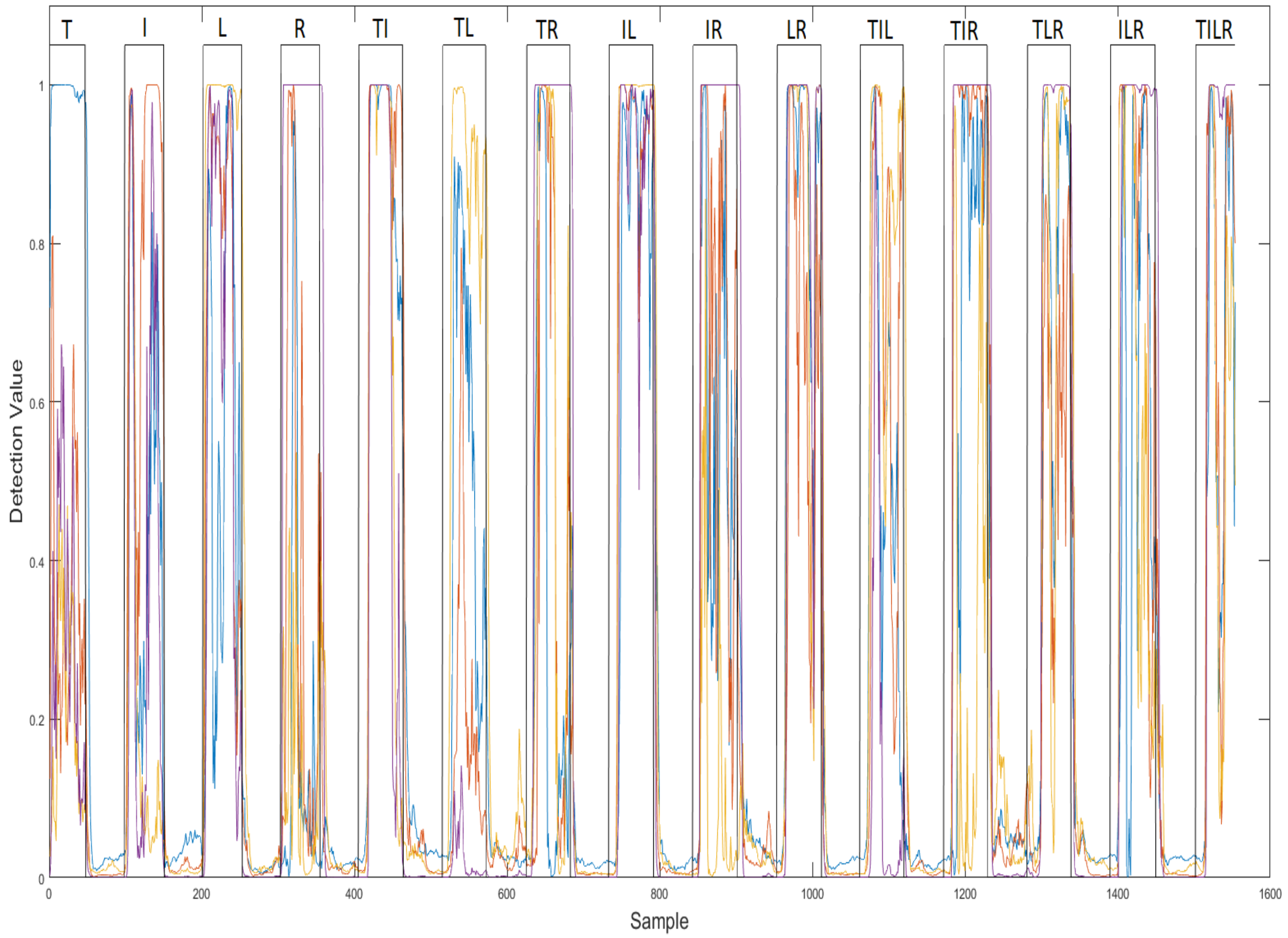

Figure 30: Results from a MLNB classification on Myo data with the arm placed in the stand. The black line together with the letters shows the correct label set. Values above 0.5 indicate a detected label according to the method.

4 fingers were used in this data: thumb (T)(blue), index (I)(red), long (L)(yellow) and ring (R)(purple).

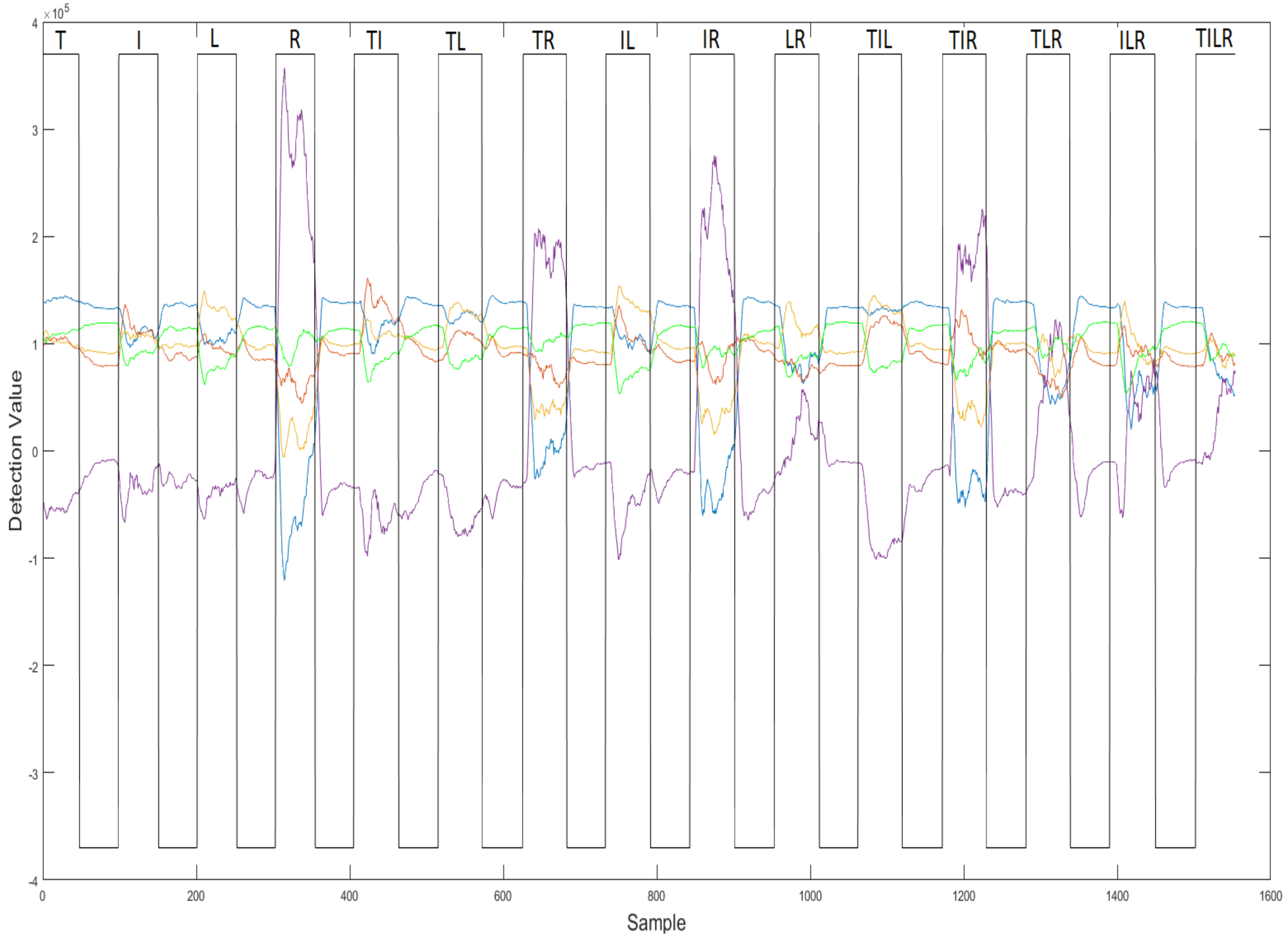

Figure 31: Results from a Rank-SVM classification on Myo data with the arm placed in the stand. The black line together with the letters shows the correct label set. The green line gives the threshold values above this indicates a detected label according to the method.