867H1: Wearable Technologies

Candidate No.
218831

Project Assignment

June 5, 2020

Contents

Chapter 1

Introduction

In a single sentence, wearable technologies are intelligent devices capable of being worn by the user. These devices are constantly expanding to the mainstream market of consumer electronics in the form of wearable assistants and fashion accessories, and the most common purpose of for these relies on the medical and healthcare subjects as fitness trackers and hearing-aid devices. Although wearable devices have gained significant popularity in the last years, these have been in the scope of researchers and consumer electronics manufacturers for a few more decades. Devices such as Ed Thorp's casino roulette predictor [1] and Steve Mann's wearable computer [2] are examples of the early stages of wearable devices; these devices were conceived with different purposes, as Thorp's device had a specific goal to achieve while Mann's was a general-purpose computer.

One remarkable feature of these devices is being *Context Aware*, meaning that the device is aware of the current situation of the user. A situation is referred to the action being currently executed, the location and, in some cases, the user's intention. The research in context awareness is highly correlated to Machine Learning, due to its intention to classify what is the user doing and experiencing; hence, context-aware devices use Machine Learning algorithms to achieve the previous goal.

Machine Learning is a subfield of Artificial Intelligence that uses computational algorithms to turn empirical data into usable prediction models [3]. As stated before, Machine Learning is heavily used in wearable technologies to classify sensor data into situations in order to make a context-aware device. Machine Learning algorithms require data as a source of knowledge, and this data can be labelled into classes that represent situations for the model to properly predict, this is commonly known as *Supervised Learning*. Many algorithms such as the Nearest Centroid, the kth-Nearest Neighbors, Decision Trees and the Naive Bayes classifiers use different approaches to achieve the classification model, but all of these are still considered Supervised Learning due to their need of labels.

1.1 Project Outline

The project assignment for this module is divided into two sections. The first parts consist of a hypothetical design of a context-aware wearable device by describing the functionality to classify, the required sensors the device would require for classification, the required signal processing, the data acquisition and testing methodology.

The second part of this report consists of the design of a classification system using MATLAB. The functionality being classified is the shape a person makes moving its arm, and the training/testing data set is given in the project outline section in Canvas. Sections on sensor selection, feature selection, dimension reduction and classifier model training are included.

Chapter 2

Context Aware Device Design Concept

As previously mentioned, a hypothetical design of a context-aware device is included as the first section of this report. The design conceptualized in this section is not meant to be constrained to a specific industry or purpose, meaning that design parameters as cost, performance or usability are not a limitation; instead, they were discussed on how different products would need different considerations. Inevitably, the concept design is often compared to a smartwatch fitness tracker since these are widely available in the consumer electronics market and may have several of the functionalities, features and sensors described in the following sections. The only assumption this concept does es is that the device is wearable and must be attached to any part of the body without limiting the user's actions, so it must be battery-powered, not invasive and relatively small.

2.1 Functionality Description

The functionality being for classification is to track the user's actions; this includes when the user is in walking, exercising, sleeping or idle. The previous actions were selected because the hypothesis is that classifying actions using ambiguous and poorly specific labels will make the design process more manageable while keeping the result useful. This functionality is commonly available in mid and high-end smartwatches and is commonly associated with several other characters tics on the device. Fitness trackers and smartwatches use this functionality to do different things such as enabling the performance tracker while exercising, disabling the push notifications while sleeping and counting steps while walking; hence, these categories are used as status flags by the device's processing unit. Although these characteristics are commodities, there is a growing market of fitness tracking technologies in the form of hardware and software to get and process the data.

In order to classify all these activities, differences between these must be acquired. These differences must be related to the physical and physiological characteristics of the user and its environment; the physiological conditions are of great importance because, it is inferred that this device will be attached to the user's body; hence, the user's condition can be considered in the classification process.

One obvious way to know when the user is performing specific activities is movement, as is a handy way to distinguish between three primary levels of activity: idle, subtle movement and exercising. While exercising, the user is expected to do fast repetitive movements such as running, punching and lifting; the activities considered as exercise are limited to the ones that may be done at home, a GYM or outdoors. Walking involves moving at a constant velocity with some vertical acceleration spikes; also, the arms may be moving repetitively as the user travels. The idle and sleeping behaviour are very similar, both are expected to have subtle movements, but the sleeping behaviour may keep the user's body in the same position for hours, while the idle behaviour may include some actions like typing, gaming, eating, and more.

As previously mentioned, the physiological condition of the user is also meant to be considered as the user's body react to performed actions. Exercising have several results in the human body, such as increasing the body temperature, increasing the blood flow to muscles, sweating, increasing the heart rate, increased breathing rate, and more [4]. Similar effects are found while sleeping such as lowering the

blood pressure, lowering the breath rate, lowering the body temperature, and more; but these effects are present in different stages of sleep after 90 minutes asleep [5]. The remaining actions do not have effects as noticeable as the two previous actions, but a benchmark heart rate and body temperature must be provided in order to perform the classification. Basic information about the user (Height, weight, age) must be provided to normalize the physiological conditions and to make a generic classifier valid for every human.

2.2 Sensor Selection

Sensors are used to interpret physical phenomena as an electrical signal in order for a microprocessor to read it. For this application, there are multiple physical phenomena and different sensors that might be used to track how their magnitude change over time; discussion about why include or not different sensors and their advantages are included in the following sections.

2.2.1 Movement Tracking

Not every "movement related" sensor is useful to classify all the actions. For example, a GPS sensor will deliver the absolute location of the user, the absolute movement vector and the location tag (after processing) of the user, so they can be used as features to track the walking behaviour from the other, but the velocity alone is not enough to decide between the idle, sleeping and exercise behaviours; if the user does all of these activities indoors, it would not be capable of detecting the movement behaviour of the user. Other sensors as a piezoelectric sensor or a gyroscope would have different classification capabilities and can even be combined in order to get a better classification, but at the same time including several sensors may impact the processing timing and power requirements.

As of the movement behaviour, the accelerometer sensor is considered well suited to classify most of the actions. Acceleration can be correlated to the three previously mentioned primary levels of activity as significant spikes in acceleration (depending on the nature of the action) can be expected from exercising, subtle changes may be related to general movement, and long periods without acceleration (Not considering gravity) may be related to the idle state. Nonidealities of accelerometers can make classification much harder. Since the relative velocity and position can be estimated using the acceleration profile over time, any noise in this profile would add destructively making the estimation inaccurate to the point of being worthless. Equation 2.1 shows the ideal relation between acceleration, velocity and position, while equation 2.2 shows how noise can affect the overall estimation of velocity and position using the acceleration profile, although it mostly depends on the signal to noise ratio.

$$x(t) = \int v(t)dt = \iint a(t)dt \tag{2.1}$$

$$x(t)+t^2n = \int [v(t)+tn]dt = \iint [a(t)+n]dt \tag{2.2}$$

Since the design is not constrained to a single sensor, multiple of these can allow different combinations, and the end product might not use all the tested sensors; differences can be present in the design prototype and end product if the design cycle allows that. Although design constraints exist that might include cost, battery life and performance, and these may affect the usage of different sensors.

Many smartwatches now include a GPS tracker for navigation and better workout-tracking, but most of the devices that include this sensor are high-end products, so the user would have to pay for better results. This sensor would allow classifying actions as previously mentioned while also provide the baseline for other non-related functionalities.

Also, a second acceleration sensor might significantly improve the detection capabilities of the device, resulting in a chest-wrist combination. Having two acceleration sensor is expected to ease tracking walking and exercise routines that keep one element static while the other is performing a movement such as push-ups and bench-pressing. The combination of sensors will be assumed to be implemented in a way that satisfies the initial requirement of non-invasion.

2.2.2 User Condition Tracking

The previous combination of sensors may be efficient for classifying the three primary levels of activity, but the sleeping behaviour is still not adequately sensed so it can be distinguished from the idle behaviour. In the previous section, it was stated that exercising and sleeping can change the physiological condition of the human body, so any of the previously mentioned conditions could be measured in order to distinguish between asleep and awake behaviours. Not every condition is easily measured, as the breath rate, blood pressure and blood flow involve more invasive sensors than the requirements allow; hence, the most accessible variable to monitor would be the heart rate, since there is a wide range of sensors with different approaches and costs.

The body temperature sensor was considered, but as the body temperature is greatly affected by the ambient temperature, and it is slowly affected by exercising and sleeping, the inclusion of this sensor was discarded. Also, a real-time clock was considered but discarded because it would be used to provide sleep quality tracking instead of sleep tracking alone, as the user can be asleep at any time of the day and this device is not expected to correct sleep habits.

The two methods for heart rate monitoring are based on different principles: electrical and optical. Electrical heart rate sensor measures the voltage generated by the expansion and contraction of the heart chambers; these sensors are commonly known as Electrocardiograms (ECG) and are widely used in medical equipment due to their accuracy and reliability.

An alternative sensor solution to ECG is the bioimpedance (BIM), which estimates tissue properties, body composition and cardiovascular parameters. BIM sensors measure the skin's impedance as a response to an AC signal at two different locations. These sensors are often used in the newest medical equipment and placing the sensors is often done by the health care staff. [6, 7]

As of the optical heart rate sensors, these use light to illuminate the capillaries in the skin so the reflected light can react to the blood vessels being pumped with blood, allowing the device to calculate the pumping rate which translates to heart rate. These sensors are the most widely available in con-

Figure 2.1: Wearable ECG.

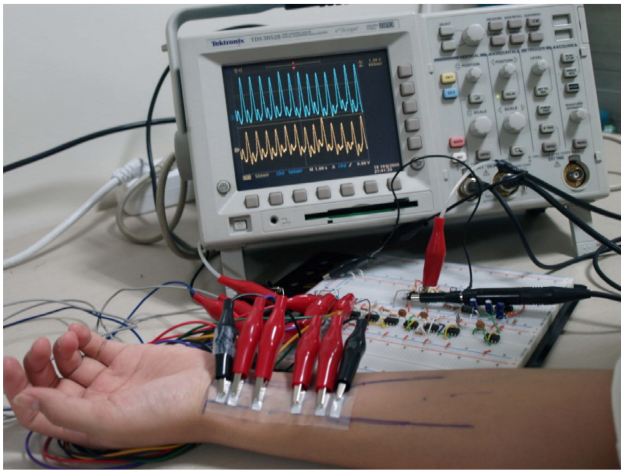

Figure 2.2: BIM sensor prototype. [7]

Figure 2.3: Optical heart rate sensor in a Fitbit smartwatch.

sumer market, as no particular steps are needed for the user to place the sensor, and its accurate enough to track sleep patterns. [8]

Although all these sensors may provide useful data to do the intended classification and have been used in large-scale consumer products before, their performance, cost and placement difficulties may be considered design constraints. The BIM sensor may be considered as the one with the most features, but the hardest to place; ECG is the most mature of the three, but wearing it may be unpractical and expensive; lastly, the optical sensor may be the least accurate but is the easiest to use for the average user, and its performance has shown it is well suited for long term monitoring. [8]

Two widely used methods for sleep tracking are Polysomnography and Actigraphy, both having very different approaches, requirements and overall performances. Polysomnography involves tracking the waves produced by the human brain in order to determine when the user is sleeping by using several electrodes on the scalp [9]. Polysomnography is much more accurate than Actigraphy, and it can track the different sleep stages, but at the same time is highly inconvenient due to the electrodes. Actigraphy is the more convenient method since it only evaluates the overall movement of the body and can be done with an accelerometer. Although actigraphy is a very similar approach like the one presented for activity tracking, as it tries to use the sleeping movement pattern of the user to classify sleep instead as the user might move while sleeping.

2.3 Signal Processing

Signal processing refers to the pre-processing the sensor's signal may have and the feature selection in order to classify the actions. The main difference between these two stages is that pre-process prepares the data coming from an experiment for it to be summarized into a single number by using a feature. As a summary of the selected sensor in the previous section, these are two accelerometers (wrist and chest), GPS and an optical heart rate sensor. Each one of these sensors needs pre-processing and different features.

2.3.1 Pre-Processing

In this stage, signals are treated to produce, enhance or keep their critical characteristics for classification away from the noise. Most of these treatments are related to filtering a portion of the frequency response and normalizing the amplitude in the time domain, but some sensors may require several stages of pre-processing and the overall output of the treatment may be a continuous or discrete signal.

The GPS module is an example of the kind of devices that require more complex processing. Since the coverage of satellites drives a GPS module, if the user is indoors the output position may not be as accurate as an outdoor measurement; also, the output location may be noisy, having a variance in the order of even tens of meters. The Kalman Filter is a popular observation technique that is often used in control system applications. It is done by imitating the measured signal by using a mathematical model based on the system (the Newton equations for this case), and other sensors (an IMU for this case) in order to get an initial condition and then use the real measurement and its noise's variance to determine the reliability of the sensor, for then outputting an estimate location. GPS modules often output their readings in latitude and longitude coordinates; the designer may keep this convention or convert it to UTM for easier post-processing. The estimated location can be used to calculate the user's velocity by differentiating the location in time.

The heart rate sensor also needs a pre-processing stage to get the heart rate value; the algorithm may vary depending on the implementation. The pro-

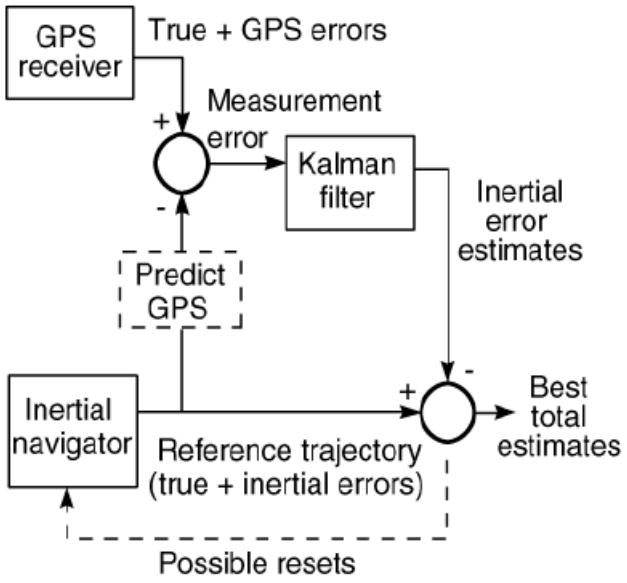

Figure 2.4: Kalman filter block diagram.

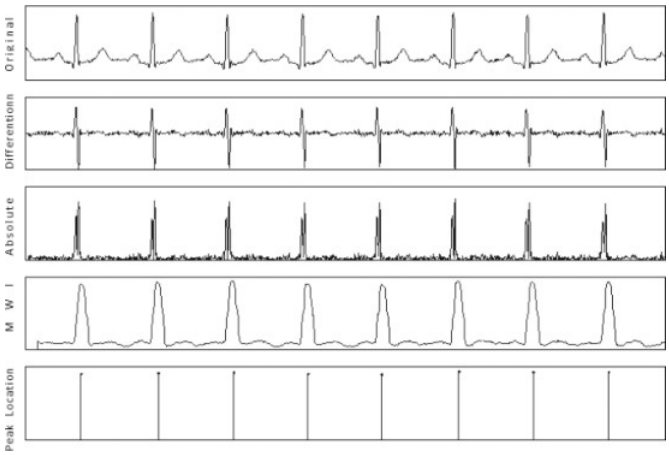

Figure 2.5: Signal processing stages in [10].

cess described in [10] uses the absolute value of the derivative of the heart rate sensor's signal; then, the resulting signal is passed through a moving integrator and then the elapsed time between peaks is calculated and interpreted as the heart rate value. Figure 2.5 shows the signal at different stages of the algorithm, as mentioned before, the pre-processing stage is required in many applications to get a usable value. Finally, the hear rate value might be normalized using data from the user such as weight, height, age and sex.

Some features related to the accelerometer do not care about the direction of acceleration nor the gravitational pull that is always present in accelerometer readings. This treatment is done by reducing the number of dimensions to a single value by using equation 2.3; this is commonly referred to as the Signal Magnitude Vector. The gravitational pull component can be removed by using a High-pass filter with a relatively low cut-off frequency [11, 12], as very low frequencies want to be measured, the filter is designed as a 2nd order HPF with a cut-off frequency of 0.5 Hz. Lastly, the signal coming from the accelerometers may be converted from bits to G's to make the readings comparable between accelerometers.

$$SMV(t) = \sqrt{x^2 + y^2 + z^2} \tag{2.3}$$

2.3.2 Feature Selection

Feature selection is probably the most critical part of the design cycle since these are the characteristic properties of the sensor's signals summarized in a single value. Features have to be selected using prior knowledge of the signals and which properties could be related to the intended labels. For example, to classify the sounds coming from a piano, the most characteristic feature is the highest peak in the frequency response. This knowledge is often the reason behind the sensor selection, pre-processing and window length selection, but in some other scenarios, the designer can come up with several other features without adding hardware or software requirements. Going back to the previous example, the Total Harmonic Distortion of a given note in a piano is different from the same note in a guitar; adding this feature would (ideally) only have costed a higher sampling frequency.

Often, more features may result in more reduced performance if only a few of these can be linked to a single category. In the intended concept design, the GPS data may be combined with an API to know the type of location where the user is at the moment. The previous is also an example of discrete features as the location type is a label, instead of a scalar. This feature can be handy for tracking when the user is exercising, as the probability of this action happening is much higher when the user's location is labelled as "GYM" than any other. However, the feature would only add uncertainty to the other actions because the probability of them

happening remains consistently low in every other location label, as not every action is linked to a location and even exercising can take place outside a GYM. A vast training dataset with many location labels would be required to turn this feature into something useful.

Previous studies have used different features to classify a single action. Work in [13] use the average peak values and average elapsed time between peaks of acceleration as features to detect exercise, and these were valid for many types of indoor routines involving repetitive movements (Lifting). Walking detection was achieved in [14] by using the autocorrelation of the acceleration measurements using a device attached to the waist. An actigraphic solution is described in [11] which applies the convolution operator to the accelerometer signal and an LTI impulse response (A low-pass filter) and then compared the output to a threshold.

Even though all these features can be tested using a feature extraction algorithm, these algorithms have their limitations. It is known that filter selectors as the Mutual Information (MI) filter are well suited for getting the efficiency of individual features but does not evaluate the combined performance of multiple features. However, wrapper methods train the classifier using several combinations and return the one with the highest accuracy; this might take time, as the number of features and the dataset length can impact the processing time. The wrapper method must iterate through $2^M - 1$ combinations, where M is the number of features. The computer used for the practical implementation in chapter 3 was able to process between 150 and 300 combinations per second using around 30 and 5 features, respectively. Using a list of 15 to 20 features and a constant estimated processing rate of 200 combinations per second to estimate the elapsed processing time is shown in equation 2.4, where R is the processing rate, M is the feature number, and the output is given in minutes. It can be noticed that time is doubled when a new feature is introduced.

$$T_p = \frac{2^{\mathbf{M}} - 1}{60 \times R} \tag{2.4}$$

$$\mathbf{M} = \begin{bmatrix} 15 & 16 & 17 & 18 & 19 & 20 \end{bmatrix}, R = 200$$
$$T_p = \begin{bmatrix} 2.73 & 5.46 & 10.92 & 21.84 & 43.69 & 87.38 \end{bmatrix}$$

The most significant advantage of using the wrapper method to select the best combination of features is that the designer may try and guess several features; the only constraint is the processing time previously discussed.

1. Average maximum value of acceleration (wrist).
2. Average minimum value of acceleration (wrist).
3. Average elapsed time between upper peaks in acceleration (wrist).
4. Average elapsed time between lower peaks in acceleration (wrist).
5. RMS value of the auto-correlation of acceleration (chest).
6. RMS value of the derivative of the GPS location.
7. Average value of the heart rate.
8. Average value of the acceleration (wrist and chest).
9. Maximum value of the auto-correlation of acceleration (wrist and chest).
10. Integral of the auto-correlation of acceleration (wrist and chest).
11. Standard deviation of the heart rate.
12. Standard deviation of the acceleration.

The candidate feature list may contain the previously listed operators, and these consider the preprocessing methods previously discussed. The first five elements on the list consist of features used in [13, 11, 14]. These features have worked in other applications and may work for this one, but no knowledge is found on their combined performance. However, these are considered as the most likely to work. Elements six to eight are the proposed features resulting from the analysis in 2.1 and 2.2, these features are expected to be amongst the selected features. Elements nine and ten are the designer's

hunch, as the work in [14] used the autocorrelation function to get the non-randomness of the acceleration profile. As some of the actions may consist of periodic movements such as walking, running and lifting the autocorrelation function may be useful to classify those actions; while the idle and sleeping behaviours may more random or noisy. These features are experimental and may work, but some experimentation may be required for them to be amongst the best feature combinations. The remaining features are some standard statistical operators on the processed sensor signals; these features are not expected to work, but if the added load is not yet compromising the wrapper timing, the features can be added for experimentation. By this point, some other standard features such as the kurtosis, skewness, frequency power spectrum, zero-crossing, fundamental frequency and more can be tested by replacing the least useful features in the list.

The wrapper function will output the best combination of features in terms of accuracy, but the following top combinations should also be considered. A comparison of the best combination and the second-best should be made highlighting the performance difference and cost difference of both combinations as they contain features from different sensors. For example, a combination having a resulting accuracy of 95% may be more valuable than a combination having 96% if the first combination uses fewer sensors (saving power and cost) or the classification time is lower than the second. As the design constraints change, so does the cost-benefit ratio, and some compromises must be made to enhance another characteristic.

Other techniques to estimate the feature's performance, such as LDA and PCA, will be described in chapter 3, as they are mostly used for data visualization of multi-dimensional feature plots.

2.3.3 Window Size Selection

The previous sections discussed the hardware required to get the dataset and its treatment to create a classifier model, but no discussion on the characteristics of the dataset. The dataset samples must be obtained considering that the device is battery powered, so reducing the on-time of the sensors may result in significant life improvement. Sample window is a collective term found in Machine Learning and refers to the lapse in which all the sensors gather data in order to it to be pre-processed and then used to feed the classifier model; this concept consists of two main parts, window size and jump size.

The window size refers to the period in which sensors are gathering data. This lapse must be large enough for the sampled signals to have the necessary characteristics and even redundancies in order to classify the action, but not that large for the signal to include samples from another activity or to result in a power spike. As this parameter is highly related to the nature of the classified labels, values may very different depending on the application, and some experimenting can be useful to get the optimal performance. Again, the approach of [13] uses a window size of 5 seconds to detect exercises activity, while they work in [14] uses 10 minutes, and even a continuous sampling method can be used for sleep tracking as seen in [11] due to the long battery life of their test device. The significant discrepancy between window sizes do not mean there is no correct answer; instead, it is a constraint that the window size may be similar to the lowest lapse, being this value 5 seconds.

The window jump size refers to the lapse between sample windows. Again, this lapse must be large enough for the device to save some power, but short enough for not losing valuable data. As the previous research did not intend to develop a monitoring device as the one presented in this report, the is no information about window jump sizes, but an estimation can be done by choosing the size to be smaller than the elapsed time to make the classification label obsolete. For example, if the user is sleeping then it is expected to remain asleep for the next 10 minutes; while walking may need to be updated every 30 seconds or even sooner.

The combination of these two parameters has been proved to impact the classifier's performance, so a careful selection process is needed to get the optimal performance. As stated in before, the minimum windows size and jump sizes are 5 seconds and 30 seconds, respectively; this combination would be accurate for tracking exercising, walking and idle states, being sleeping the only state in which the device's resources would be wasted by sampling so frequently. The proposal given in this report is hav-

ing two classifiers, one of them using the most accurate feature combination, and the other the next best combination that requires the least amount of sensors. Both of these classifiers must be accurate enough to track sleep accurately, but the secondary classifier is allowed to have a lower accuracy for the remaining activities or not consider them at all. The reason for this is to have a high-performance classifier capable of tracking every action while awake, and when the user is confirmed to be sleeping in several sample windows, the classifier will switch to the secondary classifier, which will disable the additional sensors and modify the sampling window to a larger window size and window jump size. When the user wakes up, and the secondary classifier detects the user is not sleeping, it will switch to the primary classifier again. This proposal needs to define what will be the switching conditions and the sampling window conditions when using the primary and secondary classifiers.

Classifier	Primary	Secondary
Window Size	5 seconds	30 seconds
Jump Size	30 seconds	10 minutes

Table 2.1: Sample window parameters for both classifiers.

The previous table shows the proposed sample window parameters, and the switching conditions are based on what the different classifier detects. If the primary classifier detects a sleeping behaviour for 20 consecutive sample windows, the classifier will be switched to the secondary; and once this classifier detects a single non-sleeping behaviour, it will switch to the primary classifier again. This method allows the device to save energy while keeping the detection performance, and in the case of a malicious non-sleeping behaviour while using the secondary classifier, the system would only have to spend 10 minutes using the primary classifier. The parameters used in the secondary classifier may be changed to compensate for accuracy issues. For example, the window size and jump size could be set to 20 seconds and 5 minutes, respectively, while switching from secondary to main after two consecutive non-sleeping sampling windows.

2.4 Data Acquisition and Testing

This section describes how data is going to be acquired by describing the desired scenarios for training and the proposed performance metrics. As described at the beginning of this chapter, the device is intended to be a wearable that does not affect the user's ability to do the daily routine, and its purpose is to classify the user's actions through the day. Obtaining the training dataset and proposing the testing environment must be done by considering the nature of the device, as it is intended to work for extended periods uninterrupted (Ideally non-stop).

The proposed acquisition method is that the training dataset must be acquired from the user doing several activities, but only recording the sensor data when the action is happening, and the user's physiological condition has been affected by the action. In the case of the sleeping behaviour, the dataset might be significantly enhanced if the data is recorded and labelled while using a Polysomnography environment or at least confirmed by a third party who is not asleep. This method might improve the classification of actions while compromising the accuracy at the transient between activities.

Many datasets for every action must be considered as the end-users may perform the actions slightly different from the test user. For the walking datasets, different walking speeds, steepness (stairs, hills), environments (indoors and outdoors) without raising the heart rate to a level technically considered as exercising by just walking; as the typical fitness tracker can consider walking slightly fast or in a steep as exercising, which is correct but not useful for the training dataset. The exercising behaviour must be recorded in a variety of different scenarios, as the work in [13] show that different exercises result in unique acceleration profiles. Exercise routines such as jogging indoors and outdoors, lifting, jumping, push-ups and more should be recorded including the scenarios discarded in the walking datasets, being these the ones having the user walking with an elevated heart rate, as it is considered as exercise. The sleeping behaviour is not as demanding as the past actions, but some specific scenarios must be considered, such as sleeping in different positions and inside a moving vehicle.

Finally, the idle behaviour recording must include activities that can be done while sitting, such as gaming, using a computer, driving, meditating and general procrastination.

Doing individual recordings and labelling before performing them is much more convenient than recording what the test user did through the day and then labelling the actions. All these recordings must be done by several adult test subjects, varying age, weight, height and sex, and the number of recordings per label per condition must be recorded at least ten times.

As this device is intended to be used for the most part of the day, the proposed testing method is to let the testing subject wear the device and to provide a way to check and confirm the current and past estimations. These datasets might be combined with the testing data if not every dataset was used in training.

2.5 Conclusion

From this point, the design workflow ends, and the only requirement is to run the feature selection wrapper and use the result to train the actual model. As for the classification method, these have their core differences, but they are all constrained to the quality of the features; hence, several classification models can be trained using different approaches. Finally, the best classifier is chosen using the testing dataset. This conceptual design shows the process of though a Machine Learning engineer might experience, and many other applications might have a much more complicated process.

Chapter 3

Classification System Design

This section describes the actual design of a classification system by implementing the contents reviewed in the lectures and applying a similar approach to what has been described in chapter 2. The content in this section is intended to be more technical, and application-focused, rather than a general implementation as the previous part described.

The following contents describe the intention of the classification system, the provided dataset, data selection, pre-processing, feature selection, model training and resulting models.

The main MATLAB script used in this section was mostly written from scratch, as some sections as the feature selection wrapper, MI, decision boundary plotter, NCC classifier trainer, feature to PDF and FDA scripts were obtained from the lab files and then modified accordingly. All the scripts are contained in a repository in the .

3.1 System Description

The system is designed in this section is a gesture classifier. This system must be able to detect a figure drawn in the air from five different options. The classification must be achieved using the Machine Learning principles described in chapters 1 and 2 by training four different classifiers (KNN, NCC, Decision Trees and Naive Bayes) using a variety of features obtained using filter selection and dimension reduction methods. The design process of this classifier is backed by the information obtained from the data selection stage to the feature selection stage, but some trial and error were used to get better performance as described in section 2.3.2.

3.2 Sensor & Dataset

The provided dataset was obtained from the Human Activity/Context Recognition website and consisted of data coming from 8 different tri-axial accelerometers. The dataset is sorted into different layers as a MATLAB cell array type accessed by typing `dataset{sen}{gest}{exp}` in MATLAB's environment, where `sen` is the axis number from the 8 sensors (1:x_1, 2:y_1, 3:z_1, 4:x_2, 5:y_2, ...) as shown in figure 3.1, `gest` is the type of gesture and `exp` is the experiment containing the readings from the previous sensor and gesture. Each gesture has 53 experiments with an individual length from 550 to 750 samples; as the sensors were configured to sample at a rate of 96 Hz, the sampling window size was set to around 6 seconds.

The gestures are often numbered, and a particular colour was assigned to each gesture as many kinds of plots represented the data at different stages.

1. Triangle Up
2. Square
3. Circle
4. Infinite
5. Triangle Down

3.3 Data Selection

The first step is to examine the dataset so a comparison can be made of all the sensors. As the accelerometer sensor has three individual measurements, and there are eight sensors, a total of 24 individual signals are contained inside the dataset with

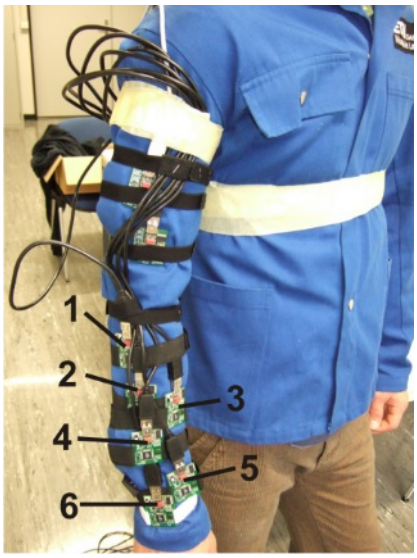

Figure 3.1: Wearable sensor array. [15]

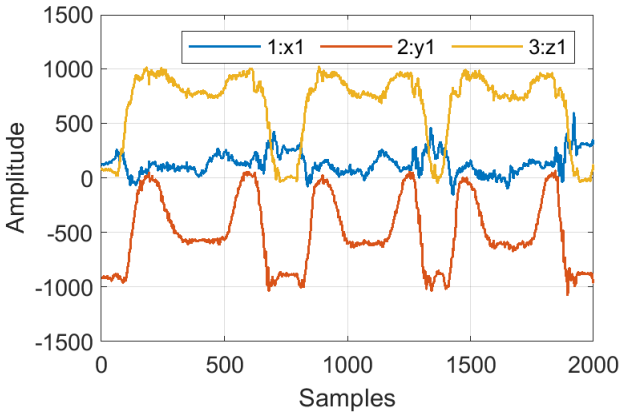

Figure 3.2: Sensor 1 signals from three different experiments for the Triangle Up gesture.

53 experiments. The amount of data this dataset provides is enormous, so a visualisation strategy was used to see the differences between sensors. The strategy is just to reduce the dimensions by combining all the experiments together to make the data accessible by choosing two layers, gesture and sensor. A short script was written to make this reduction and display the result in MATLAB's signal analyser.

The first thing to notice from figure 3.2 is that the acceleration profile from sensor 1 (indices 1, 2 and 3 in the dataset) include the gravitational pull as it remains unprocessed, this is why all three plots are shifted vertically. As of the waveforms, it can be seen that the x axis plot does not have a unique shape as the other two axes do and remain constant

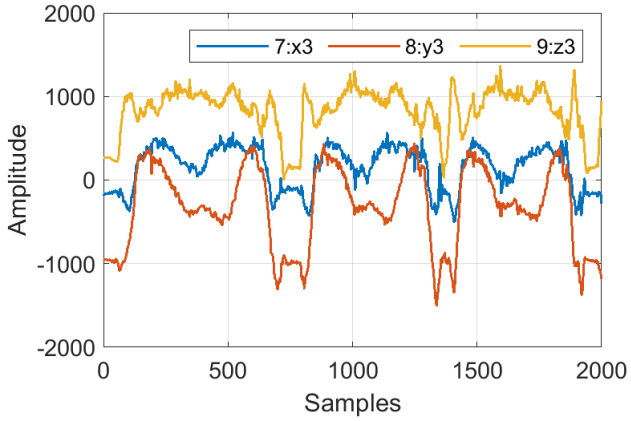

Figure 3.3: Sensor 3 signals from three different experiments for the Triangle Up gesture.

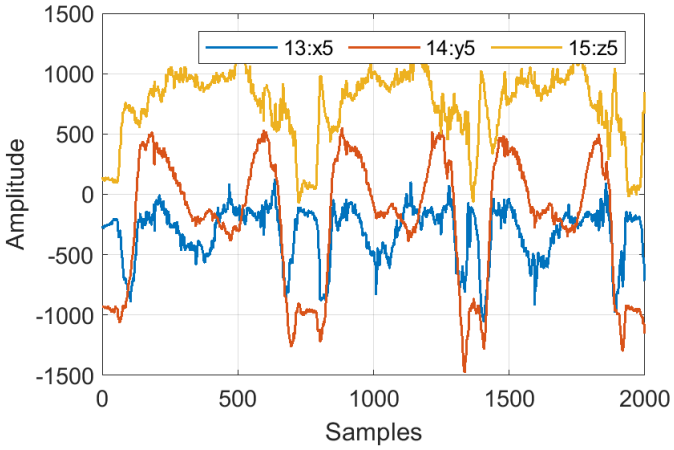

Figure 3.4: Sensor 5 signals from three different experiments for the Triangle Up gesture.

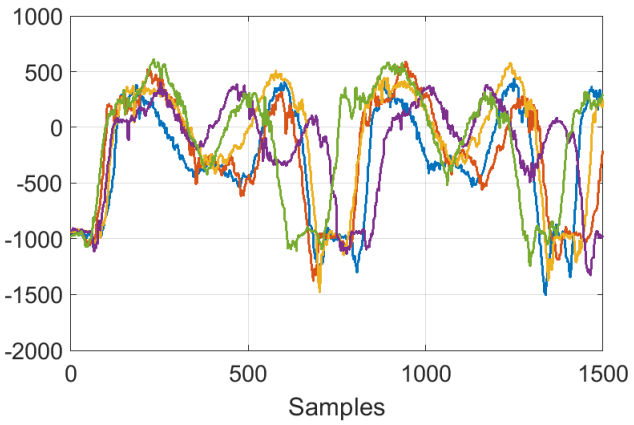

Figure 3.5: y axis acceleration profile for all gestures.

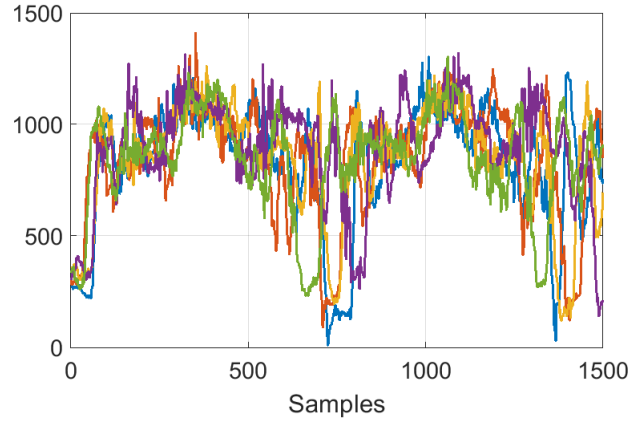

Figure 3.6: z axis acceleration profile for all gestures.

mostly through all three experiments, meaning that the x axis of sensor one do not provide as much information. Compared to sensor 3 in figure 3.3, the x axis seems to be more active, while the other two remain with a similar profile compared to sensor 1. This information shows that, when the arm is extended, x axis in sensors 1, 3 and 5 are parallel, and since the arm must not be extended entirely in order to draw the figure, the human body kinematics pull sensor 3 and 5 further faster than it does to sensor 1; meaning that sensor distance to the shoulder is proportional to its sensitivity and this affects all axis of the sensors. This property can be used as an advantage in data selection.

By comparing the single-axis acceleration profile for all gestures shown in figure 3.5 it can be seen that the y axis remains similar through all gestures excepting the Infinite gesture, which has three peaks instead of two. Including this axis may be useful to isolate the Infinite gesture from the others. The z axis shows the most variation between gestures, as no pattern between the five is visible in figure 3.6. A pattern between gestures may help to isolate them.

As a result of the data analysis, the following sections describe the analysis of the classifier design using sensor three and how is it compared to a classifier using other sensors, as the remaining sensors are believed to be redundant. Sensor three was selected as the target of the design because it shows a mid-level of sensitivity as it is placed in the middle of the lower arm; then the analysis can be replicated into other sensors.This work hypothesises that sensor sensitivity due to the human kinematics greatly enhances the classifier's performance but does not improve after a given distance.

3.4 Pre-Processing

The pre-processing stage in this design is minimal as many processes were applied to the dataset to highlight the useful characteristics. Three processes were applied, but only a single one of these presented an improvement over the unprocessed performance.

The first process is normalisation, which consists of shifting the data to make its mean zero and reduced the overall spread. Normalization is done by using equation 3.1. The resulting accuracy using normalisation was lower overall when combined with the features in 3.5; this was proved to be true for both the normalised dataset and normalised features.

$$x_{norm} = \frac{x_n - \bar{x}}{\sigma_x} \tag{3.1}$$

The second pre-processing method was a high pass filter to remove the gravitational pull component of the accelerometer signals. Combining the signals from the three axes by using equation 2.3 is recommended to get slightly better results, but the overall results were not as accurate as of the unprocessed signals. A hypothesis for the previous behaviour is that the candidate feature list relied on the non-normalized and DC components of the gravitational pull as it shifts the signals depending on the sensor orientation.

The final processing method was to analyse the signal in the frequency domain by using the FFT. This process results in an array of the same size of the time-domain array, each element representing a bandwidth of f_s/N Hz, where f_s is 96 Hz and N is the number of elements in the array. This process enables the use of many other features. For example, a candidate feature is to measure the average frequency response but only starting from the third element of the FFT; this results in a similar feature as the average magnitude without considering the DC component. Using this process was not as useful for the initial candidate feature proposal, but it was useful for the optimised candidates.

3.5 Initial Candidate Features

As stated before, the MATLAB script for designing the classifier is a modified version of the one provided in the lab files as this one allows to select data from different sensors instead of analysing a single axis. After a few test using this script and many iterations of trial and error, a list of optimal candidate features was obtained. Before describing the best features candidate list, a summary of the iteration process is described in the following paragraphs.

At first, an extensive list of candidate features was used to test the classifier's performance by using the trainer wrapper method for the Naive Bayes as the primary performance metric as it consistently had the best result compared to the MI filter and Decision Tree wrapper. At first, a resulting accuracy of 95.45% was achieved by using the y axis of sensor three alone and a list of 9 features shown in the following list.

1. RMS of the frequency response.
2. RMS of the frequency response above $f_n = 3$.
3. RMS.
4. Standard deviation
5. Median
6. Kurtosis
7. Skewness
8. Integral window
9. RMS of the difference.

Then, the feature selection wrapper was intended to be executed using the data from all three axes of sensor 3, but that resulted in a total of $2^{9\times3} = 2^{27} = 134217727$ combinations, giving a total processing time of 186 hours by estimating using equation 2.4 setting $R = 200$ and $M = 27$. The feature list was meant to be reduced using performance metrics such as the MI filter, LDA, PDF and the wrapper, being the latter the last to be used due to the processing time required.

The MI filter listed the individual contribution of each one of the features, this alone cannot be used as a performance metric, but the least useful features had an entropy reduction much lower than the top features. The sorted MI features are given in the following list, showing that the kurtosis and full frequency response magnitude are the less useful features.

1. Median (1.45)
2. RMS (1.14)
3. Skewness (1.13)
4. Integral (1.06)
5. Standard deviation (0.84)
6. Frequency magnitude above the 3rd FFT element (0.69)
7. Kurtosis (0.61)
8. Frequency magnitude (0.53)

The LDA dimension reduction acted as the first metric of feature combination, as the gestures were to be separated by the algorithm; the more isolated the clusters, the better. The PCA reduction is not as useful as only shows the correlation between the classes, but a performance metric can be obtained by evaluating the spread of the clusters. As seen in figure 3.7, the LDA analysis show some overlap between gestures, meaning that the features are not entirely separating the gestures.

The Probability Density Function (PDF) was used to if a feature could distinguish each one of the

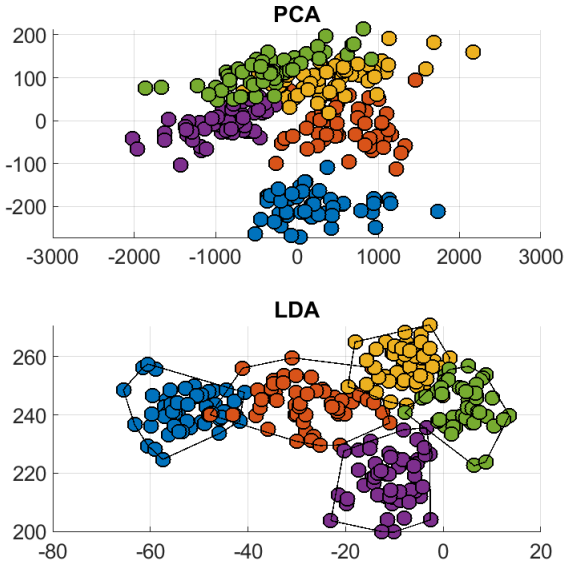

Figure 3.7: PCA and LDA analysis of the initial proposal.

gestures; as the classifier uses a combination of features, not every single one of these must be able to separate the gestures, but if any of these can isolate different gestures then the expected performance of the classifier is increased. Figure 4.2 (3 Appendix) shows that many features present overlapping gestures, making them harder to classify. Again, the most distanced gestures are the Infinite gesture, being this the easiest to classify. Although the MI filter depends on the PDF, it fails to show the which feature combinations might be useful.

The overall performance of these features is shown in table 3.1 using the top 5 best feature combinations from the Naive Bayes' feature extraction wrapper; the tables use the numbering as shown in the candidate feature list, earlier in this section.

Combination	Accuracy
2,4,5,9	95.45%
2,3,4,5,9	95.45%
2,4,5	94.69%
2,3,6,9	93.93%
2,4,5,6,9	93.93%

Table 3.1: Top 5 combinations for the initial features set using the Naive Bayes wrapper.

The confusion matrix was useful to find which features did not help to differentiate from another

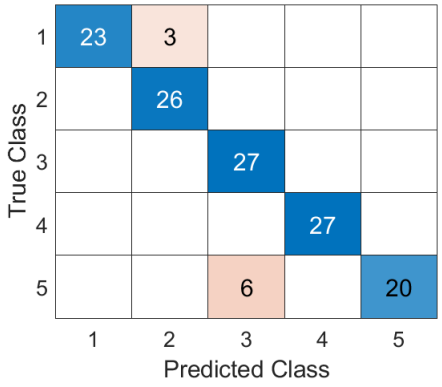

Figure 3.8: Initial confusion matrix for every feature using the y axis of sensor 3.

specific feature. Figure 3.8 shows that three samples classified as Square (2) are in reality Triangle Up (1), and six samples classified as Circle (3) are in fact Triangle Down (5). As expected, the Infinite (4) gesture is the easiest to classify.

3.6 Optimized Candidate Features

After using the previous method to reduce and replace the candidate feature list and some experimentation, the optimised candidate features resulted in the following list.

1. Frequency magnitude above the 3rd FFT element
2. Standard deviation
3. Median
4. Mean absolute deviation
5. Skewness

By reducing the candidate feature list and replacing the most irrelevant features, it is possible to use data from several sensors, simultaneously. The feature performance analysis shows many differences between this and the previous iteration of candidate features; the following analysis shows the effect of the five candidate features applied to all three axes of sensor 3 (15 features total).

The MI filter sorted the candidate features us-

ing their performance and showed that features that the y (S8) axis provides the most information as every feature using this axis was in the top half and the z (S9)axis provided the least information as discussed in section3.3. Besides, some features that were ranked low in the initial proposal now are in the top as the Frequency Magnitude. Replacing the kurtosis by the median absolute deviation was a vast improvement since it gives more information than the kurtosis.

1. S8: Median (1.45)
2. S7: Frequency Magnitude above $f_n = 3$ (1.27)
3. S8: Skewness (1.13)
4. S7: Standard deviation (0.96)
5. S8: Standard deviation (0.84)
6. S8: Mean absolute deviation (0.83)
7. S7: Mean absolute deviation (0.74)
8. S8: Frequency Magnitude above $f_n = 3$ (0.69)
9. S9: Mean absolute deviation (0.66)
10. S9: Standard deviation (0.61)
11. S9: Median (0.59)
12. S9: Frequency Magnitude above $f_n = 3$ (0.44)
13. S7: Skewness (0.36)
14. S7: Median (0.36)
15. S9: Skewness (0.23)

The dimension reduction techniques in figure 3.9 show that the gesture clusters distanced to the point that only the outlier samples of the cluster are overlapping with other clusters.

The PDF for the optimised in 4.3 (3 Appendix) benefits from the increased number of data sources and optimised features showing that the x axis (S7) is efficient to track the Triangle Down in the frequency domain, as not any other feature in all three sensors can distinguish that gesture as good as this. The median absolute feature in the z axis (S9) isolates all the gestures far enough to be a selected candidate. Finally, highly ranked features in the previous proposal as the skewness were not as efficient using data from multiple axes, as the skewness in the z axis fails to isolate the gestures excepting the Triangle Down .

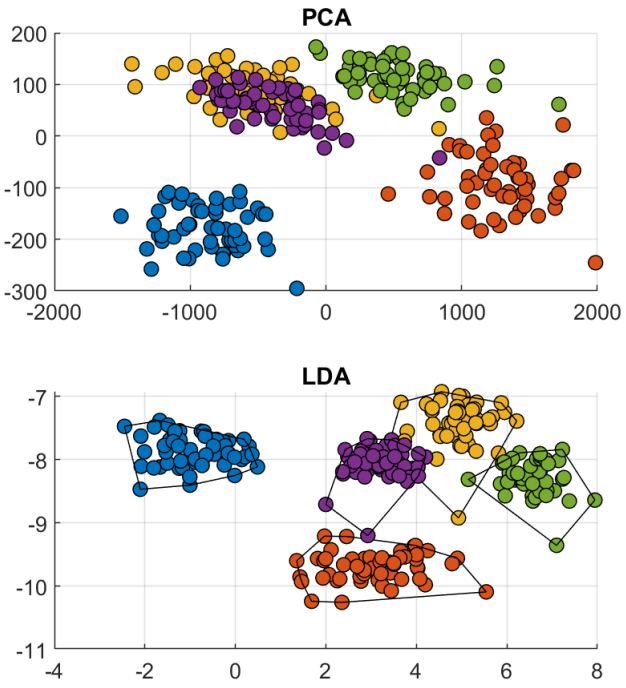

Figure 3.9: PCA and LDA analysis of the optimized set.

3.7 Performance Discussion

Reducing the feature list made running the feature selection wrapper much more accessible, as 15 features coming from the three axes of a single sensor were processed in only three minutes. After using the Naive Bayes wrapper to get the best combinations among the optimised feature set, the top 5 combinations were evaluated as shown in table 3.2 using the numbering of the optimised feature list in section 3.6. This set of features are considered the end result and its performance are described in the following paragraphs.

Table 3.2 shows that the top 5 feature combinations highly outperform the top feature combination of the first candidate features, even having two combinations with a 100% accuracy. This improvement can be confirmed by the confusion matrix in figure 3.10 as it shows a perfect score for all gestures.

In order to get a 2D representation of the deci-

Combination	Accuracy
S7:1,2 - S8:2,3,4 - S9:3,5	100%
S7:1,4 - S8:2,3,4 - S9:3,5	100%
S7:1,4 - S8:1,2,3 - S9:3	99.24%
S7:1,2,4 - S8:3,4 - S9:3	99.24%
S7:1,2,3 - S8:1,3,4 - S9:3	99.24%

Table 3.2: Top 5 combinations for the optimized features set using the Naive Bayes wrapper

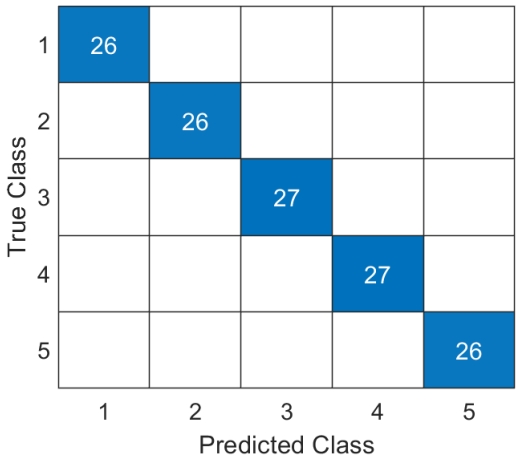

Figure 3.10: Confusion matrix for the best feature combination in the optimization set.

sion boundary, a reduced feature set given by the LDA algorithm was used. The LDA algorithm gets a combination of new axes that project the gesture clusters as separated as possible. The linear transformation produced by these axes can reduce the dimension of the feature set, packing the features into a linear equation represented as a matrix multiplication as shown in equation 3.2, where w is the linear transformation matrix given by the LDA function.

$$x_{LD} = x_{features} w \tag{3.2}$$

$$w = \begin{bmatrix} -0.000052018 & -0.0015006 \\ 0.022276 & 0.021595 \\ -0.029326 & 0.0018565 \\ 0.018326 & 0.0015189 \\ 0.03599 & -0.0032525 \\ -0.0029108 & -0.005534 \\ -0.9985 & -0.99974 \end{bmatrix}$$

The decision boundaries in figures 3.11, 3.12, 3.13, 3.14 show the regions each classifier consider

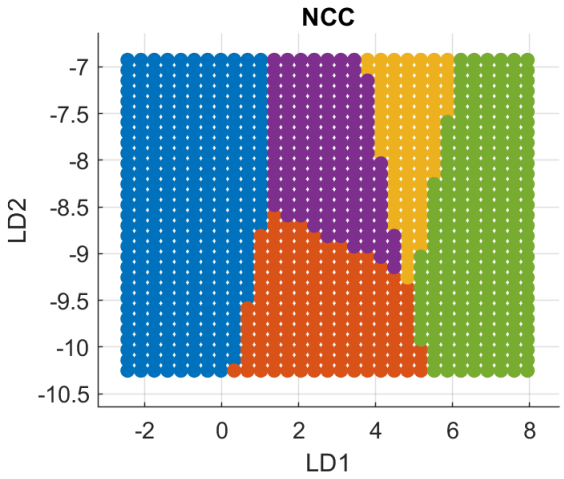

Figure 3.11: Decision boundaries for the NCC classifier after LDA reduction.

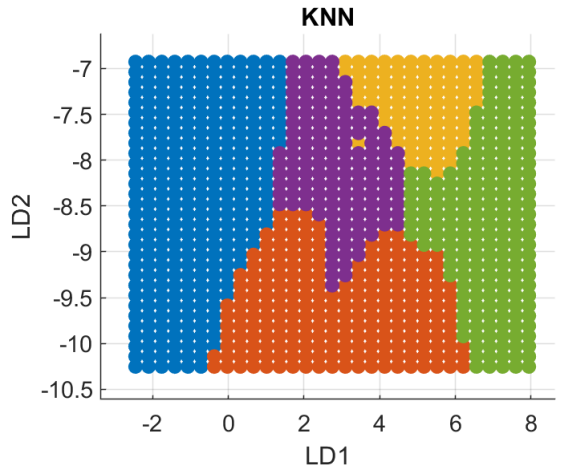

Figure 3.12: Decision boundaries for the kNN classifier after LDA reduction.

to be mapped to a gesture. Although each classifier has its particular algorithm to calculate these boundaries, it can be seen that the gestures (Triangle Up , Square , Circle , Infinite and Triangle Down) remain roughly constant between classifiers and matches the cluster position in the LDA analysis in figure 3.9. Unfortunately, it is difficult to find an interpretation of the decision boundaries related to the features as they are reduced, and each axis corresponds to a combination of features.

As the LDA reduction introduces a new parameter in the classified system, the accuracy of every classifier is slightly reduced in comparison to the multi-dimensional classifier. Table 3.3 shows the accuracy of both optimised classifiers compared to the initial proposal.

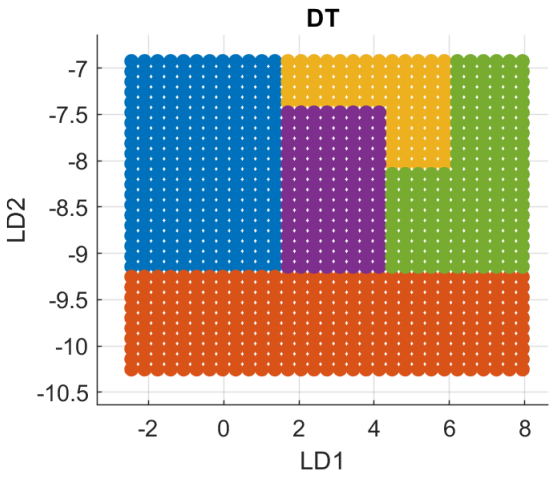

Figure 3.13: Decision boundaries for the DT classifier after LDA reduction.

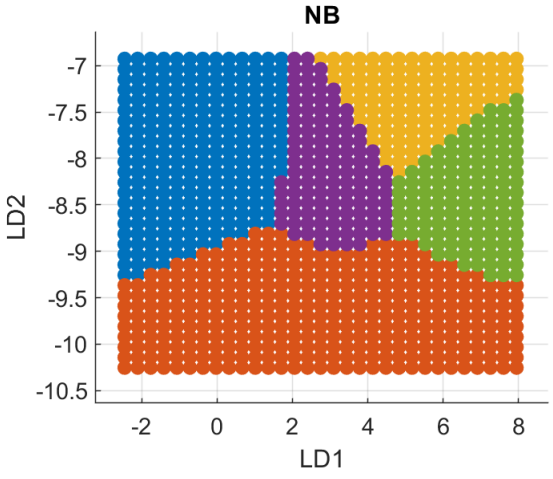

Figure 3.14: Decision boundaries for the NB classifier after LDA reduction.

Classifier	NCC	kNN	DT	NB
Optimized	73.48	93.93	92.42	100
Reduced Opt	94.69	93.93	94.69	97.72
Initial	51.51	75.00	87.87	95.45

Table 3.3: Comparison of different classifiers

Another performance metric is the processing speed and depends on the complexity of the classifier. For example, the decision tree pictured in figure 4.1 (4 Appendix) shows the number of conditional statements that are required to reach a decision and that the classification of the Triangle Up (1) gesture needs far less conditional evaluations than any other gesture. Although a classifier in a wearable device would be executed in an embedded processor, the speed performance can still be compared using a computer.

A test environment was created using the test dataset by repeating it 1000 times and measuring average processing time per prediction. The results state that the fastest classifier is the decision tree with 0.007 μs per prediction, as it only has to do four conditional evaluations at the most. The Naive Bayes comes as second with 0.019 μs per prediction, showing that its improved performance does not imply any compromises in speed. The kNN comes as third with 0.037 μs per prediction, and finally, the NCC is the slowest by much with 2.38 μs per prediction.

Finally, the performance of the same feature set is compared with the data coming from every other sensor using the Naive Bayes wrapper. Table 3.4 shows that the optimised feature set is efficient with every sensor's measurements.

Sensor	NCC	kNN	DT	NB
1	72.27	91.66	91.66	99.24
2	59.09	62.87	94.69	99.24
3	73.48	93.93	92.42	100
4	75.75	92.42	96.21	100
5	59.84	87.12	88.63	98.48
6	68.93	93.93	93.18	100
7	86.36	93.18	90.90	100
8	53.78	84.04	87.87	100

Table 3.4: Comparison of the classifiers using different sensors and the optimized feature set

3.8 Conclusion

Many things can be discussed from the design of this classifier. As half of the dataset was used as testing data, it may have included the outliers shown in figure 3.9 and only samples close to the cluster's centre were used as test data, explaining why a 100% accuracy was achieved with most sensors. An example of an outlier affecting the decision boundary of a classifier can be seen in figure 3.12, where a Circle dot can be seen inside the Infinite region.

The initial hypothesis made in section 3.3 was proved wrong in table 3.4, as there is a noticeable performance improvement in sensor three that remains constant in the remaining sensors. Sensor 7 and 8 are not labelled in figure 3.1 and were inferred to be the two sensors close to the shoulder; these sensors had a similar performance to sensors closer to the hand. This analysis might mean that every sensor in this array, in some way, have diagnostic information from the gestures, since not the same combination was selected the best by the wrapper.

Although it is believed that pre-processing gives some kind of improvement, the results from the pre-processing stage for this design had the opposite effect. This behaviour must be related to the nature of the features.

As the initial feature selection was mostly inspired by the features proposed by the lab files, the shapes and patterns found in section 3.3 were summarised as a single number using the features. This is mentioned because, as a human could recognise the shapes and patterns with the required training, the Machine Learning algorithm used simple statistical functions to classify the gestures.

If this system were to be implemented in an embedded processor, the easiest classifier to port would be the Decision Tree, as only consists of a few `if` statements; a greater effort is required to port any of the remaining classifiers.

Not every step of this design process was as deeply analysed as described in chapter 2, as this was an example of an adequately acquired dataset and the vast availability of data, and the primary purpose was the design of a Machine Learning algorithm.

It was fun! :)

Chapter 4

Appendix

4.1 Repository

The MATLAB scripts used in the making of this classifier are available in the following repository.
`https://github.com/JoseAmador95/UoS_Wearable_Technologies`

4.2 Figures

These figures are referred in chapter 3, and describes the Decision Tree visual model and the PDF graphs for the gestures (Triangle Up , Square , , Infinite and Triangle Down).

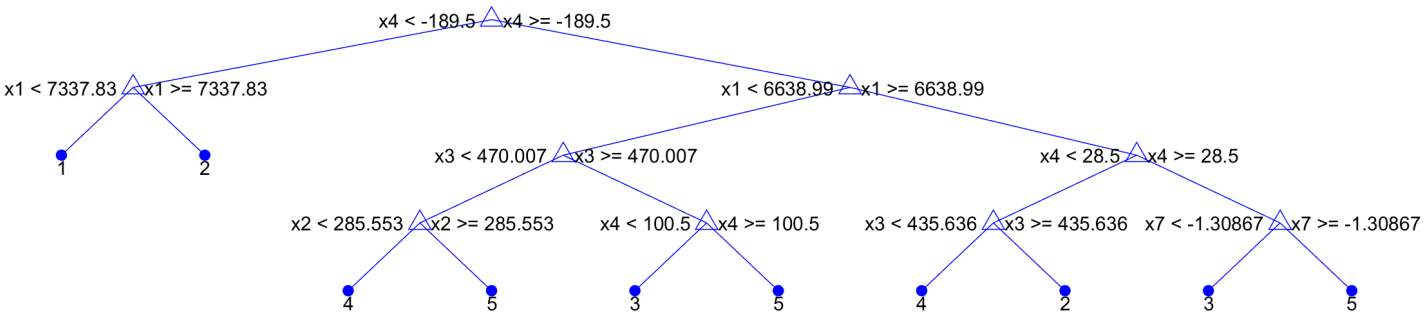

Figure 4.1: Visual diagram of the decision tree using the top combination from the NB wrapper.

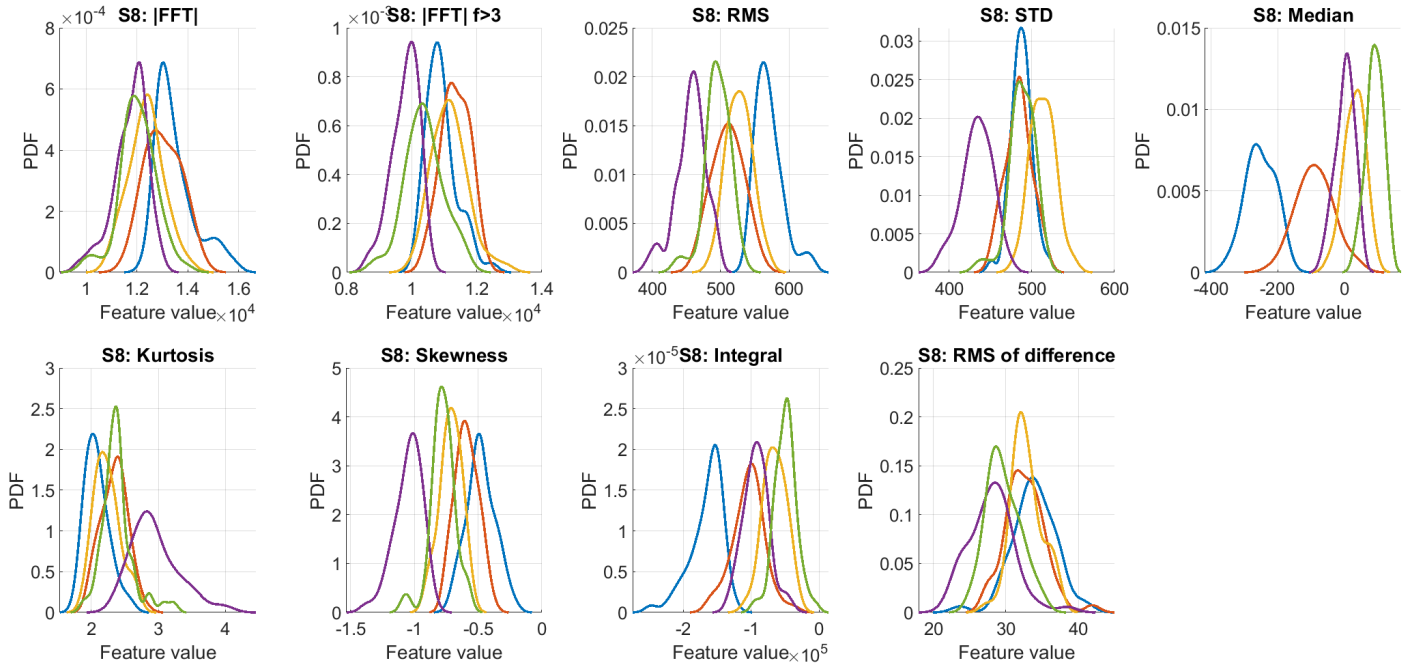

Figure 4.2: PDF from the initial feature set using the y axis of sensor 3.

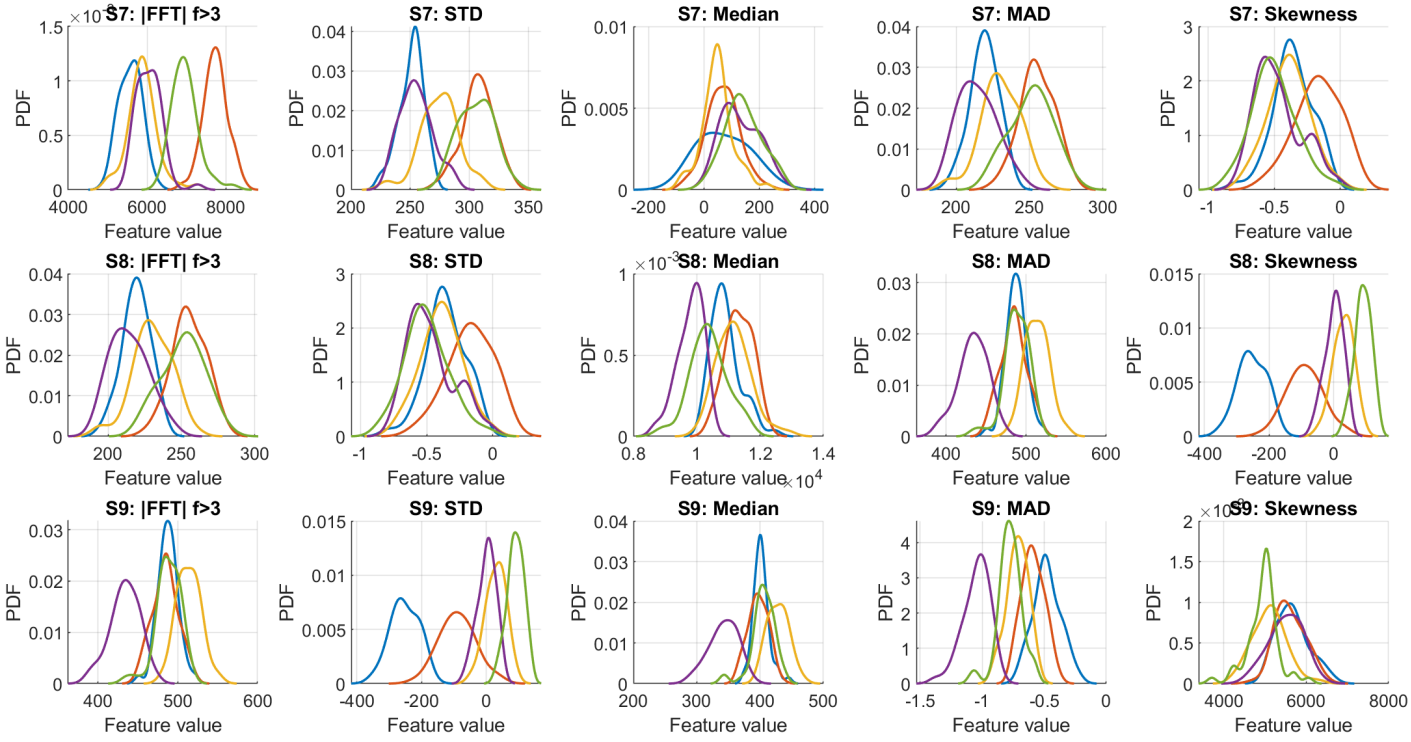

Figure 4.3: PDF from the optimized feature set using all axes in sensor 3.

Bibliography

[1] E. O. Thorp, "The invention of the first wearable computer," in *International Symposium on Wearable Computers, Digest of Papers*, vol. 1998-Octob, pp. 4–8, 1998.

[2] S. Mann, "Smart clothing: The wearable computer and WearCam," *Personal and Ubiquitous Computing*, vol. 1, no. 1, pp. 21–27, 1997.

[3] G. Shobha and S. Rangaswamy, "Machine Learning," in *Handbook of Statistics*, vol. 38, pp. 197–228, Elsevier B.V., 1 2018.

[4] "Short term effects of exercise on the body systems - Long and short term effects of exercise - Eduqas - GCSE Physical Education Revision - Eduqas - BBC Bitesize."

[5] National Sleep Foundation, "Sleep eBook," 2018.

[6] M. C. Cho, J. Y. Kim, and S. H. Cho, "A bioimpedance measurement system for portable monitoring of heart rate and pulse wave velocity using small body area," in *Proceedings - IEEE International Symposium on Circuits and Systems*, pp. 3106–3109, 2009.

[7] D. Naranjo-Hernández, J. Reina-Tosina, R. Buendiá, and M. Min, "Bioimpedance sensors: Instrumentation, models, and applications," 7 2019.

[8] D. Phan, L. Y. Siong, P. N. Pathirana, and A. Seneviratne, "Smartwatch: Performance evaluation for long-term heart rate monitoring," in *2015 International Symposium on Bioelectronics and Bioinformatics (ISBB)*, pp. 144–147, IEEE, 10 2015.

[9] B. Beth-Cooper, "How does your fitness tracker know when you're asleep?," 2014.

[10] E. A. P. J. Prawiro, C. C. Hu, Y. S. Chan, C. H. Chang, and Y. H. Lin, "A heart rate detection method for low power exercise intensity monitoring device," in *2014 IEEE International Symposium on Bioelectronics and Bioinformatics, IEEE ISBB 2014*, IEEE Computer Society, 2014.

[11] J. Chuan, Z. Sheng, and L. Xiaokang, "An Effective Way to Improve Actigraphic Algorithm by Using Tri-axial Accelerometer in Sleep Detection," in *2014 IEEE 17th International Conference on Computational Science and Engineering*, pp. 808–811, 12 2014.

[12] V. T. van Hees, L. Gorzelniak, E. C. Dean León, M. Eder, M. Pias, S. Taherian, U. Ekelund, F. Renström, P. W. Franks, A. Horsch, and S. Brage, "Separating Movement and Gravity Components in an Acceleration Signal and Implications for the Assessment of Human Daily Physical Activity," *PLoS ONE*, vol. 8, 4 2013.

[13] C. Lee, "Movement detection and analysis of resistance exercises for smart fitness platform," in *International Conference on Ubiquitous and Future Networks, ICUFN*, pp. 410–415, IEEE Computer Society, 7 2017.

[14] J. D. López, A. Sucerquia, L. Duque-Muñoz, and F. Vargas-Bonilla, "Walk and Jog Characterization Using a Triaxial Accelerometer," in *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pp. 1406–1410, 10 2015.

[15] D. Roggen, "Human Activity/Context Recognition Datasets."