



UNIVERSITÀ POLITECNICA DELLE
MARCHE

FACULTY OF ENGINEERING

Master of Science in Biomedical Engineering

Heart Rate Prediction from Vowel Speech Signals Using Machine Learning Techniques

Adviser: **Prof. Stefano Squartini**

Coadviser: **Dr. Florian Eyben**

Candidate: **Luca Alberto Pettinari**

Academic Year 2018/2019

Abstract

In this thesis, it is shown that heart rate can be predicted from audio with good accuracy, using classification and regression algorithms on a large group of features extracted from vowel speech signals by the openSMILE toolkit, a benchmark in large space feature extraction. To this aim, a dataset containing synchronized audio/ECG recordings from 31 subjects is established for audio-based heart rate prediction. The experimental setup is tailored for obtaining a homogeneous set of labels, distributed evenly on a range of about [60, 140] BPM. On this dataset, different feature sizes have been tested with SVM, both using linear or RBF kernels. In a speaker-independent testing, heart rate can be determined with a correlation coefficient of 83% and a mean absolute error of 10.6 BPM. Using the same set-up, low/high classification yields an accuracy of 86%. Performances are promising: however, limitations are bounded to the threshold used to assign classification labels ("Low"/"High"), and to heart rate extraction based on audio chunks, preventing real-time audio-based heart rate predictions.

Contents

1	Introduction	3
1.1	Fundamental Hypothesis	4
1.2	State of the Art	5
1.3	The Munich Biovoice Corpus	7
1.4	Aim of this Thesis	9
2	Acquisition System	11
2.1	Recording equipment	12
2.2	ECG Acquisition Protocol	16
2.3	Audio-ECG Synchronization	21
2.3.1	Beep Detection Algorithm	22
3	The BioMiner Application	31
3.1	Features and Modalities	32
3.1.1	Acquisition Mode	33
3.1.2	Analysis Mode	38
3.2	Software Architecture	41
3.2.1	The Configuration File	43
3.2.2	The <code>BioMinerUserInterface</code> Class	45
3.2.3	The <code>BioMinerSession</code> Class	47
3.3	Heartbeat Detection Algorithm	48
3.3.1	Electrocardiogram	48

3.3.2	Heart Rate Measurement	54
3.3.3	Preprocessing Stage	56
3.3.4	Adaptive Threshold	57
4	Modeling	65
4.1	Support Vector Machine	66
4.1.1	Soft-margin Decision Surface	72
4.1.2	The Kernel Method	75
4.1.3	Support Vector Regressor	78
4.2	Feature Extraction	84
4.3	Classification and Regression Models	86
5	Experimental Results and Discussion	91
5.1	Data Collection	91
5.1.1	Experimental Protocol	92
5.1.2	Dataset Description	95
5.1.3	Comparison with the MBC	107
5.2	Results	108
5.2.1	Classification	108
5.2.2	Regression	111
5.3	Discussion	115
6	Conclusions	117

List of Figures

1.1	Main vessels of the head.	5
1.2	PPT between the R-wave and the arterial blood pressure.	6
2.1	Experiment room, with: fitness cycle, widescreen to follow the ongoing acquisition and camera lights to light up participant's face.	12
2.2	Top view of an Arduino UNO Rev 3.	13
2.3	Top view of an Olimex SHIELD-EKG-EMG.	14
2.4	Sennheiser PC 8 USB-Headset.	14
2.5	Cable and electrodes.	15
2.6	Arduino and Olimex connected to the piezoelectric speaker and the 10 μ F capacitor.	15
2.7	ECG recording unit encased in a plastic box.	16
2.8	GoPro HERO5.	16
2.9	Packet format.	19
2.10	Example of a 5 s ECG signal.	21
2.11	Scheme of synchronization between ECG and audio signal.	22
2.12	Detail below the fitness cycle: ECG recording unit location.	23
2.13	Mel spectrogram of three /a/ vowels, computed over the interval $[0, 8000]$ Hz.	24
2.14	Spectrogram line from Figure 2.13 at $f_0 = 4400$ Hz.	25
2.15	Example of refinement of τ_f	27
2.16	Original waveform giving the spectrogram of Figure 2.13, now with rectangular impulses replacing beeps.	29
2.17	Scheme of the beep detection algorithm.	29

3.1	Initial screen.	33
3.2	<i>General information</i> box.	34
3.3	Example of <i>Health and lifestyle</i> form filling.	35
3.4	Screen of the acquisition mode.	36
3.5	Fulfillment of an /e/ sustained vowel with a timer (Figure 3.5a). End of the same task with description and saving options (Figure 3.5b).	37
3.6	<i>Annotations</i> box.	39
3.7	Screen of the analysis mode.	40
3.8	Class diagram of the application.	42
3.9	Waveform morphologies and segments of a normal ECG tracing (Lead I).	49
3.10	The cardiac vector $\mathbf{H}(t)$ and the Einthoven triangle.	50
3.11	Lead II electrode positioning scheme.	53
3.12	Overlapping window method to compute BPMs, with a sliding step of 1 s. Detail: poor detection performance by means of constant thresholds (in orange and green).	55
3.13	Signal preprocessing pipeline.	57
3.14	Transformed signal of 10 s long segment with fiducial marks.	58
3.15	Levels and adaptive threshold.	59
3.16	Flowchart of the <i>direct search</i> stage.	60
3.17	Flowchart of the <i>search back</i> stage.	61
4.1	Main stages of the study.	65
4.2	Three different decision boundaries for a linearly separable dataset. The black one satisfies the maximum margin principle.	68
4.3	Maximum margin separation decision boundary and support vectors.	69
4.4	An example falls within the separation margin but on the right side of the decision boundary (a). An example falls within the separation margin and on the wrong side of the decision boundary (b).	73
4.5	Non-linearly separable patterns.	75
4.6	Support vector network.	77
4.7	Regression line with the maximum margin approach.	80

4.8	ε -insensitive tube and hinge loss function.	81
4.9	k -fold cross-validation.	87
4.10	Modeling pipeline.	88
5.1	Fitness cycle detail: GoPro camera.	93
5.2	Directory tree of the dataset.	95
5.3	Example of a false positive (incorrectly detected peak) and a false negative (peak not detected at all).	102
5.4	Example of two consecutive false negatives.	102
5.5	Germany: 20 participants (in red); Iran: 3 participants (in pink); Brazil, Croatia, Greece, Ireland, Italy, Portugal, Russia, South Korea: totalling 8 participants (in pale pink).	104
5.6	Box Plot for Sustained Vowels (at rest).	105
5.7	Box Plot for Sustained Vowel at 100 BPM.	106
5.8	Box Plot for Sustained Vowel at 130 BPM.	106
5.9	Confusion matrices for the best models with linear kernel, respectively: UFS ($n = 3000$, Figure 5.9a) and WFS ($n = 250$, Figure 5.9b).	110
5.10	Confusion matrices for the best models with RBF kernel, respectively: UFS ($n = 2000$, Figure 5.10a) and WFS ($n = 500$, Figure 5.10b).	110
5.11	Hyperparameter curves for classification models.	111
5.12	Learning curves for the best models with linear kernel, respectively: UFS ($n = 3000$, Figure 5.12a) and WFS ($n = 1000$, Figure 5.12b).	113
5.13	Learning curves for the best models with RBF kernel, respectively: UFS ($n = 1000$, Figure 5.13a) and WFS ($n = 1000$, Figure 5.13b).	113
5.14	Hyperparameters curves for regression models.	114

List of Tables

2.1	Example of the output <code>.csv</code> file.	19
4.1	Three among the most used kernels in SVM problems.	78
4.2	LLDs and functionals for ComParE 2016.	86
5.1	Classification results for linear kernel.	109
5.2	Classification results for RBF kernel.	109
5.3	Regression results for linear kernel.	112
5.4	Regression results for RBF kernel.	112

Chapter 1

Introduction

The 21st century advancements of computing and communication technologies has enormously improved the chances of extracting a wide range of information of different nature from raw data. The modern means to store daily vast quantities of streaming data has produced incredibly huge databases, with a likewise amount of information locked in there, but usually not yet discovered or articulated [60]. Thus, one of the nowadays trends of information technology is to use data-driven approaches to retrieve, organize and categorize information from data. In this scenario, healthcare technologies has been revolutionized at this moment too. The variety of affordable and portable medical devices allowing a patient to actively contribute to diagnosis and treatment is permanently increasing, drastically changing the traditional model of clinical examination, and allowing single individuals to continuously monitor and collect their own physiological data out of hospitals or healthcare institutions. There are devices for measuring blood pressure, heart rate, body core temperature, skin conductance, respiration rate and many other physiological parameters autonomously, though they are still rather expensive and inconvenient for an everyday use. Ideally, monitoring of vital parameters should require minimal effort by the patient and cause minimal disturbance, and the whole process should be assisted by intuitive and easy-to-use interfacing systems. For these reasons physiological data should be recorded by sensors that are best "unnoticed" in terms of intrusiveness [48]. In particular, in emergency circumstances, where the readiness of getting even a raw measure or estimate of a vital physiological parameter is

of primary importance, or in daily monitoring, i.e. situations in which it is just required a rough estimation of the parameter in question, the high resolution provided by dedicated measurement systems take second place, favouring contactless measures by means of sensors embedded in wide-market products within everyone's reach, like smartphones or personal computers. Also, monitoring in extreme case patients, such as burn victims or premature infants, cannot disregard non-contact approaches by offered these means, not to mention that diagnoses can be attempted also remotely with patients located in far regions [35]. In this scenario, particular interest will take on those technologies that would be able to infer the subject's physiological state in a contactless way, i.e. without attaching to the patient's body a dedicated sensor to obtain a given parameter of interest, like the heart rate. Voice technologies, i.e. those technologies using audio signals obtained from voice, satisfy this prerequisite. In fact, voice is a very direct and immediate signal to tap into, and it has been proved to be rich of information about the speaker, like important biomarkers to determine the presence of pathologies (e.g Multiple Sclerosis [17], Huntington's Disease [52]), or to extract information about a particular physiological state [37].

1.1 Fundamental Hypothesis

The interest of research towards audio-based recognition of physiological parameters in this field is growing, albeit with comparably fewer efforts. The motivation of formerly little research in this field has to be sought in the difficulty of establishing a relationship between heart activity and phonation. An attempt to describe the connection between human voice and heartbeat was given by Orlikoff et al.: heartbeat accounts for approximately 0.2% to 19% of absolute perturbation of the fundamental frequency (jitter), measured on pronunciation of sustained vowels [37]. These findings can be motivated by the fact that the vocal tract runs almost parallel to the two commons carotid arteries (Figure 1.1), where, due to their proximity to the heart, pulsatile effects (given by heart pumping) subtly modify the shape of the vocal tract, thus its transfer function. The variation of it over time will correspond to variations on the voice spectrum as very low frequency changes [10]. These changes depend on the mechanical action of blood flowing through the head vessels. Therefore, the aforementioned effect produced on voice is best correlated to signals like electroglottography (EGG) and blood pressure. Unfortunately, these signals are not easy

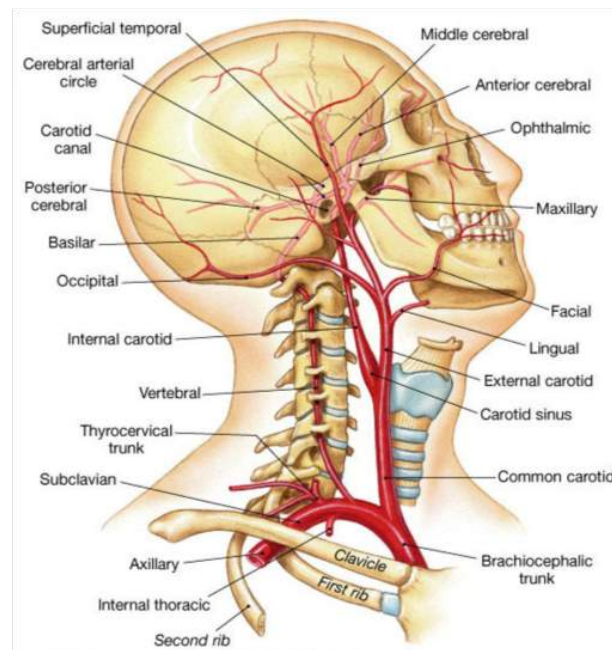


Figure 1.1: Main vessels of the head.

to collect due to sophisticated sensors needed for the measurement. Instead, sensors measuring the heart electrical activity, like electrocardiogram (ECG), which measure the variations of the heart dipole triggering heart's pumping action, are usually easier to use, cheaper and diffused in market. The downside of this choice resides in the so called **pulse transit time** (PTT), defined as the delay between the depolarization of the ventricles and the propagation of flow wave through a vessel, as shown in Figure 1.2 [54]. This depends on the physiological state of the subject and must always be taken in account when correlating ECG with the induced jitter on voice (due to blood flow).

1.2 State of the Art

Recently, different authors proposed to define methodologies to extract features correlated to such modulation effect, in particular, related to heartbeat, which represent the best recognisable event of heart activity, even in mild noisy conditions. One research trend avails mainly of time-frequency signal processing techniques. Mesleh et al. have developed a method of retrieving heart rate from short sustained vowels involving STFT and statistical order filters: the method

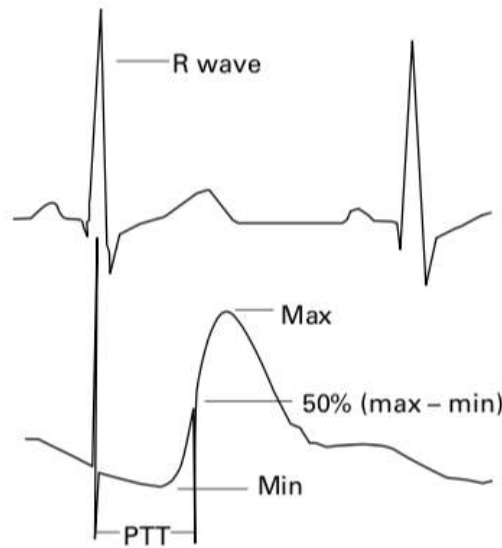


Figure 1.2: PPT between the R-wave and the arterial blood pressure.

has showed a low percentage error as opposed to heart rate estimates provided by pulseoximetry and proved to be robust in noisy conditions [35]. Yet, this method aims to extract only one single value and deals with shorts chunks of recording, preventing the study of subject's heart activity over longer periods. Ryskaliyev et al. explained a method for estimating heart rate from human speech using speech features based on Mel-Frequency Cepstral Coefficients (MFCC) [43]. Other methods take advantage of existing processing tools for feature extraction and then apply machine learning techniques for regression and classification. In speech and audio processing, a milestone of large space audio feature extraction is the open source software openSMILE [15]. As shown by Schuller et al., by means of openSMILE large space feature extraction in combination with Support Vector Machine (SVM), heart rate can be classified from sustained vowels as low or high, as well as estimates of heart rate and skin conductance can be retrieved via regression algorithms like Support Vector Regressors (SVR) [48]. Other machine learning approaches proposed the estimation of heart rate using the Hybrid Feature Vector (HFV), obtaining high performance scores for regression analysis [2]. Whilst much of the research in this field has been focusing on the use of sustained vowels, still little effort has been put in reproducing the same results when using free-speech or conversation samples. Recently, an empirical regression model has been proposed to classify speech recordings in three emotional states (joy, neutral and angry)

and correlated to heart rate changes. The results are promising when emotion are not combined together, while performance decreases as the user mixes multiple emotional states [43]. Another research work shows heart rate prediction with Random Forests (RF) using spontaneous speech coming from users interacting with a dialog system, explicitly designed to induce frustration, in order to establish a correlation between physiological response to stress situations [53]. Provided two groups – i.e. a group made of frustrated users, and a control group – the effect on heart rate of speaking itself was assessed, showing an increase of it as the user is involved to speak right after a silence period. This increment happens regardless the emotional state of the user, but the acceleration of the rhythm results more pronounced for the frustrated group.

1.3 The Munich Biovoice Corpus

The Munich Biovoice Corpus (thereafter MBC) is a dataset containing speech audio signals from participants, recorded alongside with physiological parameters, namely heart rate in term of Beat Per Minute (BPM) and Skin Conductance (SC). The equipment used to obtain those was:

- a Zoom Q3Hd camcorder equipped with an X-Y HD microphone to record audio ("room microphone") at a sampling rate of 92 kHz in uncompressed PCM-wave format;
- a Logitech Clearchat Headset to capture close-talk speech recordings;
- a Wild Divine Inc.'s "iom" device to record BPM and SC data from three sensor's attached to a subject's fingers.

These signals were recorded in a synchronized way with the audio signals, in two different physiological states: low load state and high load state after the participants had performed physical exercise. To obtain the corpus, 19 subjects (4/15 female/male, 3 Chinese, 15 German, 1 Italian) participated in the experiments and gave their consent to data recording and storage. All were free of temporary diseases, but the subjects include smokers and such with mild cardiac and neurological disorders. All completed a questionnaire about their height, weight, nationality and general health condition as well as a short personality test [41]. All subjects were recorded while breathing, while pronouncing the sustained vowel /a/ repeatedly and while reading a standard

text which is used frequently in phonetics. These recordings were performed in the two above named physical load states with both low and high pulse. Before starting the experiments, the subjects had to sit down in front of the recording equipment and they were instructed on the procedures. Then, they had to perform a practice recording session to assure they were well familiar with the procedure before the actual recording takes start. After that, the actual recording session was started. After this first step with low pulse/load, the subjects raised their physical load by exercising, i.e., running quickly up and down stairs over three floors and running through a long hallway which led to the recording room. The subjects were required to physically exercise until their pulse exceeded 90 BPM. Right after the exercise unit, step two of the ongoing recording session was performed. In each recording condition (rest/exercise), subjects were recorded sitting on a chair in front of a desk. A Zoom Q3Hd was placed 50 cm from the subjects' lips; the Logitech Clearchat headset was head-worn by the subjects. The iom's heart rate sensor was connected to the left middle finger and the two skin conductivity sensors were connected to the left ring and forefinger. The subject's left hand was electrically grounded in order to minimize the influence of electromagnetic and electrostatic noise in the sensors. For each subject, a comfortable frequency F_c for the pronunciation of a sustained /a/ vowel was determined during the practice phase with visual feedback from a live frequency analysis. This frequency F_c was marked on the screen, and the subject had to train repeating the /a/ vowel in the "comfortable" frequency as precise as possible for several times. After the subject was able to intentionally produce such a comfortable phonation within a tolerance of ± 7 Hz during 5 subsequent attempts, it was assumed that the subject could produce it reliably during the recording. The subject had to undergo the same training procedure for an /a/ vowel at a lower pitch, specified by a frequency F_l , four semi tone levels below the frequency F_c and calculated according the formula

$$F_l = \sqrt[12]{2^{-4}} F_c. \quad (1.1)$$

In each of the two recording phases, participants had to pronounce the /a/ vowel with F_c and F_l pitch, four times each. Also, they had to read a continuous text aloud. Native German speakers read out the text "Der Nordwind und die Sonne", while non-native subjects read the English translation of the text "The Northwind and the Sun". Overall, the final dataset consists

of heart rate and skin conductivity labelled audio recordings from 19 speakers. The instances are divided into 74 text periods, 644 breath periods and 630 sustained vowel expressions. They are further divided into low pulse and high pulse recordings and into headset (close-talk) and room microphone recordings. Sustained vowels are labelled with their pitch, therefore divided into sustained vowels at comfortable fundamental frequency F_c and sustained vowels in low fundamental frequency F_l .

1.4 Aim of this Thesis

In this present work, the main goal is to determine a new corpus of audio/ECG recordings to train heart rate predictive models based on voice, taking the MBC as a benchmark and introducing then innovative factors, both on the acquisition side and on the experimental design. Finally, the obtained dataset will be compared to the MBC. An ECG recording device, capable of recording the electrical heart activity synchronously to the audio track, will be needed to produce heart rate labels in terms of BPMs. To this aim, a single lead ECG recording system is provided with the experiments, in order to have more precise BPM measurements and, more importantly, to collect ECG signals, which are richer of information and more precise than pulseoximetry. In addition to this, also video-recording of the experiments will be added, whenever is possible. To coordinate all these requirements and produce data in a reproducible way, a software application will be developed in the form of a graphical user interface to favour and avail of participants' interactions during the acquisitions. Finally, classification and regression models will be obtained from the collected data to produce heart rate predictions from vowel speech signal segments, opportunely extracted from the obtained data. Indeed, results will be presented and discussed, showing promising aspects and limitations of the study.

Chapter 2

Acquisition System

In order to train models capable of extracting information about heart rate from voice, a dataset containing synchronized audio and ECG signals is needed for the purpose. In fact, synchronization of these signals is a necessary condition to study how heart activity intervenes and modifies vocal signals. In particular, it was stressed out that heart activity would showcase its effects on the audio signal in a delayed fashion, related to the physiological condition of the individual. For this reason, **synchronization** is extremely important and must be implemented delay-free per se. Such issue was already addressed [48], where timestamps synchronization between the audio tracks and BPM/SC reads from the pulse oxymeter and skin conductance sensors were aligned by means of a triggering signal (acoustic beep) at the beginning and the end of each recording. The same approach was followed in this work, with an acquisition system designed explicitly to deal with this important specific. Moreover, to simplify the recording experience both for the subject and tester, a computer application was developed to control simultaneously several processes (ECG acquisition, audio recording, heartbeat detection, etc.), to increase reproducibility of the experiments over multiple trials, and to enhance the efficiency in term of time consumption and easiness of use. The acquisition environment was acoustically circumscribed and provided with a fitness cycle to control the amount of physical exercise (Figure 2.1), opposite to a widescreen displaying the output from the graphical user interface of the application, with which the subject had to interact throughout the whole recording session.



Figure 2.1: Experiment room, with: fitness cycle, widescreen to follow the ongoing acquisition and camera lights to light up participant's face.

2.1 Recording equipment

As opposed to the great product selection of microphones to record voice, there is still relatively small choice of cheap, accessible and ready-to-use ECG recording devices. Furthermore,

they usually do not provide their own built-in module for clock synchronisation with another device. For these reasons, the choice of the ECG recording system consisted on a dedicate electronic board, containing all the the necessary circuits for signal acquisition, like buffer amplifiers, differential amplifier, and analog filters. Then, the analog to digital conversion (ADC), communication and synchronization aspects where dealt using a general purpose (GP) electronic board. The choices for this study are summarized in the following:

- an Arduino UNO Rev3, in Figure 2.2. Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software, allowing high-level firmware coding (it compiles C++). With an ATmega328P processor allowing 32 kB of program memory, its main specifics are: a quartz with clock speed of 16 MHz, an operating voltage of 5 V, 6 analog input pins and 14 digital I/O pins;



Figure 2.2: Top view of an Arduino UNO Rev 3.

- an Olimex SHIELD-EKG-EMG, shown in Figure 2.3. It is an Arduino compatible board, connecting directly to Arduino UNO analog pins. It has an AUX input connector for the electrode cable carrying the raw signal, and mounts pin sockets on top, making it stackable with additional boards. The shield is suitable for monitoring and collection of electrocardiography and electromyography data;

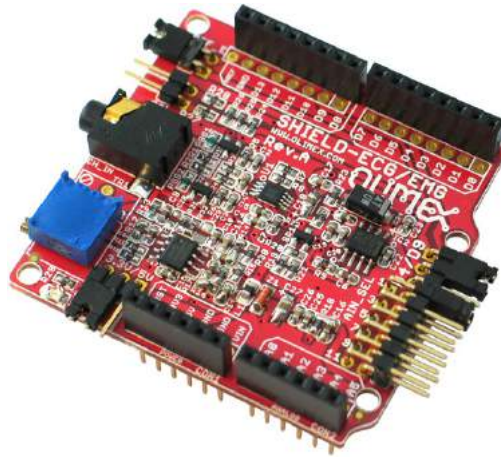


Figure 2.3: Top view of an Olimex SHIELD-EKG-EMG.

- a Sennheiser PC 8 USB-Headset, shown in Figure 2.4. Connecting with a USB plug, the headset mounts a noise-cancelling microphone with a sensitivity of -41 ± 3.5 dB (measured at a 1kHz sine wave) and a 90 – 15000 Hz frequency response range. Further, the microphone is adjustable at different heights and it was head-worn by participants always at the level of the mouth;



Figure 2.4: Sennheiser PC 8 USB-Headset.

- cable and electrodes, shown in Figure 2.5. Together, they constitute the ECG signal transducer. On one end, there is a stereo input jack with three contacts, each corresponding to a single electrode. On the other end, there are three electrode pads: the black one corresponds to the ground, the white and the red to the positive and negative poles respectively. They are stuck with disposable, pregelled, Ag/AgCl skin electrodes and they are positioned on the subject's body following a modified lead II configuration.



Figure 2.5: Cable and electrodes.

Arduino is also connected to an 80 dB piezoelectric speaker, soldered between digital pin 12 and a GND pin, and to a 10 μ F electrolytic capacitor with the positive pole connected to a REF pin and the negative connected to a GND pin, as shown in Figure 2.6.

Figure 2.6: Arduino and Olimex connected to the piezoelectric speaker and the 10 μ F capacitor.

All together, Arduino, Olimex, speaker and capacitor were encased in a plastic box (Figure 2.7).



Figure 2.7: ECG recording unit encased in a plastic box.

Furthermore, all the experiment provided also video recording under the participant's compliance, for which it was used a GoPro HERO5 Black (Figure 2.8) and camera lights to avoid darkness overshadowing subjects' faces (see Figure 2.1).



Figure 2.8: GoPro HERO5.

2.2 ECG Acquisition Protocol

To use and store ECG signals in a dataset, it is needed that the ECG recording device (i.e. the Arduino UNO plus the Olimex shield) can communicate appropriately with a PC. The first thing to do would be to investigate the sampling rate that the said device can reach. Arduino UNO has a built-in 10-bit ADC and the maximum sampling rate can be calculated from the

clock speed by the following formula:

$$f_s = \frac{\text{CLK_SPEED/PRESCALER}}{13}, \quad (2.1)$$

where the denominator indicates the number of (prescaled) clock cycles necessary for each analog to digital conversion. Given a prescaler of 128, the (2.1) allows to calculate a sampling rate of

$$f_s = \frac{16 \text{ MHz}/128}{13} \simeq 9615 \text{ Hz}. \quad (2.2)$$

The ADC conversion is called in the Arduino's main loop by the `analogRead` function and the ADC will reach the calculated conversion speed if the main loop contains only that instruction. If other instructions more than `analogRead` are present, they will take clock cycles to be executed, slowing down the calculated ADC conversion speed. As long as data are not transmitted but instead stored in memory, a wide range of sampling frequencies can be obtained varying the prescaler according to (2.1). This is done quite rarely for obvious space limitation (Arduino UNO has an EEPROM of 1024 B) and it is preferred to send the acquired signal to a storage device, like a hard-disk or a computer using serial communication. The cons of doing so is a decreased sampling frequency, no more matching with the calculations made through (2.1). Arduino is provided with UART serial communication, using the common 8N1 protocol: each byte sent through serial must be enclosed between a start and an end bit, totalling ten bits sent per byte. With a baud rate of 9600 bit/s, assuming a stream of a 8-bit wise bytes, data can be sent at a maximum of 960 Hz with the simple code snippet below.

```

#define analogPin A5
void setup(){
    serial.begin(9600)
}
void loop(){
    int sensorValue = analogRead(analogPin)
    Serial.flush();
    Serial.write(sensorValue);
}

```

The `Serial.write` instruction writes a byte to the serial port. Its execution runs in background, allowing the main loop to continue immediately after its call. This justifies the presence of `Serial.flush` function in front of it, which makes the program await for the complete transmission of the previous byte before writing a new one. Despite this, it was observed that the above piece of code does not work according to theoretical expectations, i.e. it does not allow to reach the expected sampling rate of 960 Hz. This issue is systematic and appears whenever two communicating system using UART protocol have different clock speeds [40]. Thus, the reachable sampling rate is bound by the serial communication. For this reasons, signal transmission/reception was implemented with a diverse approach, based on acknowledgement communication protocols, with the PC playing the role of a *client*, asking for ECG data samples, and the recording device regarded as a *server*. In this scheme, data will not be received in succession as a stream, but rather they will be treated singularly as **packets** and sent under specific request-acknowledgement conditions. For the sake of clarity, instead of cycling indefinitely over the `Serial.write` instruction, data is sent over the serial port when a request (consisting of a message byte) is forwarded by the client and then successfully acknowledged by the server. At this point, the server sends a packet and the client is enabled to receive it. In this study, any stream of ECG signal sent with the following approach is always enclosed between two acoustic beeps produced by the piezoelectric speaker for synchronization purposes. According to this specification, the request-acknowledge process is carried out in three following phases:

- *starting phase.* The client forwards a **beep request** (message byte: `0xFF`) to trigger the starting beep. After an ACK byte (`0xFF`) is sent back from the server to the client, which is now enabled to receive one packet, the call to the user-defined `beep()` function makes Arduino's digital pin 12 oscillate with a 4400 Hz square wave for a duration 0.5 s, producing the first acoustic beep through the piezoelectric speaker which is connected to it. The packet sent is a 4-bytes long timestamp with a microseconds precision (obtained by calling the built-in `micros()` function), and corresponds to the first rising edge of the oscillating square wave used to produce the start beep. For this reason, this instant will be regarded as the **zero-time reference value** (thereafter indicated with τ_i) of the current ECG acquisition to which all successive timestamps will be referenced;

- *data sending phase.* After the starting beep, the client sends in loop a **data request** (message byte: `0x01`). To this request corresponds an ACK, and the successive packet sending. In this case, the packet is formatted as shown in Figure 2.9: the first byte contains the first eight MSB out of the 10-bit byte obtained from a ADC conversion (paying a higher quantization error), and the resting three bytes contains the timestamp referred to the zero-time reference value τ_i . This loop lasts for a certain acquisition duration specified by the user. If this duration is not specified, the client will forward data requests indefinitely, until a SIGTERM signal is caught and the client code can exit the loop;

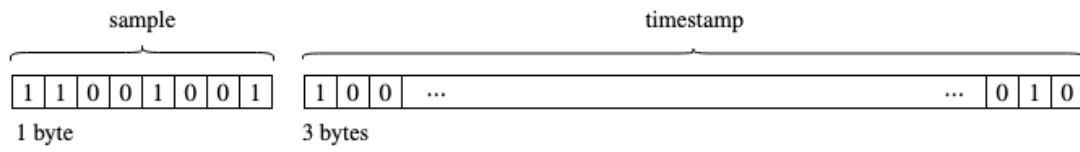


Figure 2.9: Packet format.

- *end phase.* When the client exits its main loop, it sends a **beep request** to trigger the ending beep, in the same way as happened for the starting beep. In fact, the last packet sent is also a 4-byte long timestamp with microseconds precision, corresponding to the first rising edge of the oscillating square wave used to produce the end beep. Thus, it represents the end instant of the current ECG acquisition and will be regarded as the **end-time reference value** (thereafter indicated with τ_f).

This three main parts are executed for each ECG signal acquisition, at the end of whom the received ECG is saved in a `.csv` file, which is formatted as in the table below.

k	PC_timestamps	μC _timestamps	signal_samples
0	2.532192	0.000000	0.000000
1	0.512106	0.514960	0.541176
2	0.520298	0.516044	0.541176
3	0.524292	0.521256	0.533333
4	0.528383	0.526468	0.517647
5	0.532588	0.531684	0.549020
\vdots	\vdots	\vdots	\vdots

Table 2.1: Example of the output `.csv` file.

The first column of Table 2.1 contains timestamps produced from the computer's clock, the second one contains timestamps from Arduino's clock indicating the instant when the corresponding packet was received, and the third one contains 8-bit ECG samples coming from the ADC. Therefore, the obtained signal is described by two time-series, say $t[k]$ and $s[k]$: the first one containing timestamps (second column of Table 2.1) and the second one containing ECG samples (third column of Table 2.1). This latter one in particular admits only values in a finite dynamic range ($0 - 5$ V), quantized over 256 levels, each sample corresponding to 8-bit byte (see the example reported in Figure 2.9). At last, the content of the first and last rows differs from all the others. In fact, they contain only the timestamp from Arduino corresponding to start and end beeps in their first cell, while the other two carry no information (they contain zeros). A visualization of the signal obtained using this method is represented in Figure 2.10. It is clear that the initial ramp (due to absence of sample sent) corresponds to the time interval where Arduino executes the `beep()` function, thus not sending any packet. Finally, it must be observed that any time that the client code is executed by the user for a new signal acquisition, Arduino's serial makes the board to reset by default. For the purpose, the system was provided with a $10\ \mu\text{F}$ capacitor which disables the automatic resetting of the board.

This implementation will not solve the aforementioned issue related to the sampling rate, which is intrinsically related to the characteristics of the communicating systems. Nonetheless, it has the merit to express precisely the timestamp of every sample coming from every ADC conversion. The obtained signal is though **non-uniformly sampled**. Several tests over a medium duration of 15 min long acquisition time had however shown a quite constant interval between two successive timestamps. Defining the average time interval as

$$\langle T_s \rangle = \frac{1}{N} \sum_{k=1}^N (t[k] - t[k-1]), \quad (2.3)$$

where N is the number of samples contained in $t[k]$, it was observed that the average time interval corresponded to $\langle T_s \rangle = 5.343 \pm 0.012$ ms. That implies that the signals acquired in such way can be regarded as quasi-uniformly sampled, thus all results of sampling theory can be extended to the non-uniform case. In fact, a band-limited signal with a non-uniform sampling can be perfectly reconstructed from its samples if the average sampling rate satisfies the Nyquist

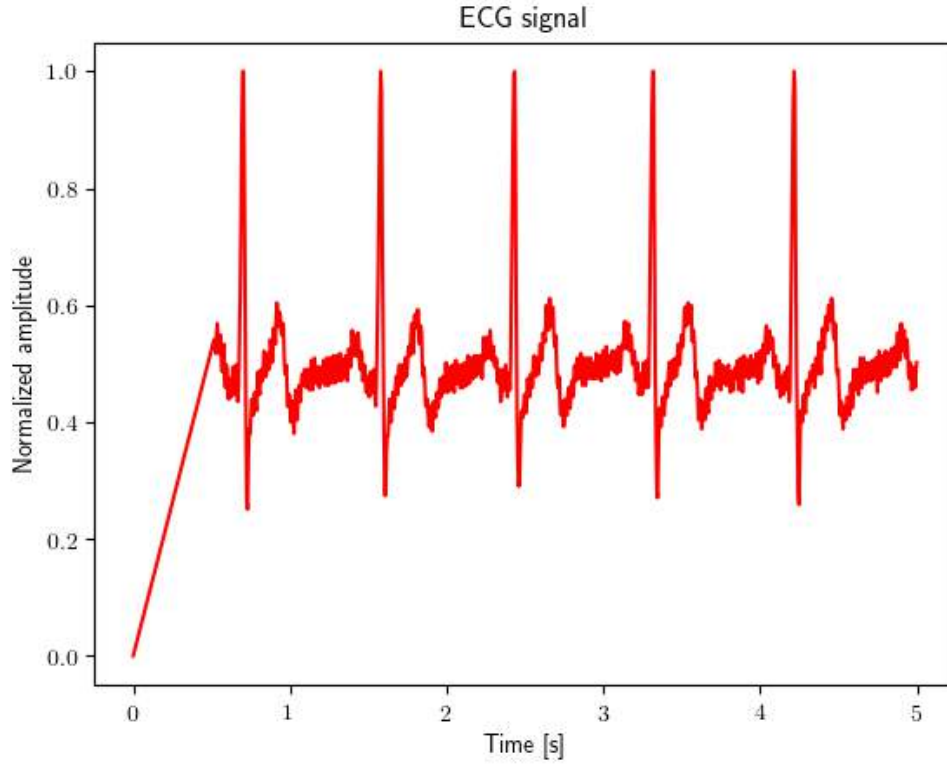


Figure 2.10: Example of a 5 s ECG signal.

condition [33]. Considering that a 50 Hz bandwidth is enough for studying heart rate variability and the most important ECG parameters [32], an average sampling frequency calculated as

$$\langle f_s \rangle = \frac{1}{\langle T_s \rangle} \simeq 187 \text{ Hz}, \quad (2.4)$$

satisfies the Nyquist inequality, being $2B = 100 \text{ Hz} < 187 \text{ Hz}$. This allows to consider it absolutely appropriate to carry all the fundamental frequency band of a standard ECG signal.

2.3 Audio-ECG Synchronization

Naturally, each ECG acquisition corresponds to an audio signal recorded by the microphone. The instruction to start audio recording is executed before and ended after the ECG acquisition

stage. Nonetheless, the timestamps τ_i and τ_f do not correspond with the start and the end of the audio signal. This mismatch is due to the implemented communication protocol, where a request must be always acknowledged before being executed by the client, meaning always a delay between request (from the client) and execution (from the server). Yet, whenever the start (or end) phase is executed, acoustic beeps are emitted just because they are almost immediately encoded on the recording audio, representing thus the rightful markers of the start and end phases of the ECG acquisition on the audio signal, as shown schematically in Figure 2.11. Therefore, in this paragraph it will be discussed the techniques employed to retrieve these two instants.

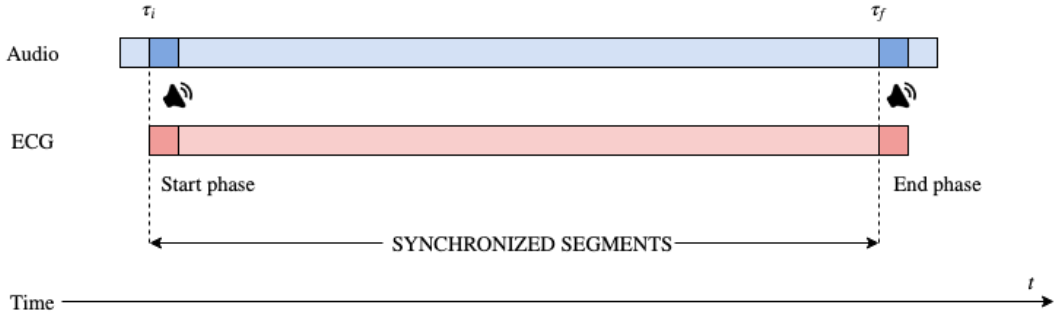


Figure 2.11: Scheme of synchronization between ECG and audio signal.

With *almost immediately*, it is intended that there is however a beep-encoding delay related to the speed of sound ($c = 343$ m/s). Considering that every participant was sitting on the fitness cycle and that the acquisition unit was located under its seat (Figure 2.12), the distance between the piezoelectric speaker (source) and the headset's microphone (sink) is in the order of magnitude of the meter, thus meaning a delay of

$$\delta = \frac{1 \text{ m}}{343 \text{ m/s}} = 2.915 \text{ ms} \quad (2.5)$$

2.3.1 Beep Detection Algorithm

To find acoustic beeps into audio recording, thus to identify both their start and rising edges, a detection algorithm was developed based on the spectrogram of the audio signal. The spectrogram is a representation of a signal in the time-frequency domain and descends directly from the **Short-Time Fourier Transform** (STFT), which is defined (in case of continuous-time



Figure 2.12: Detail below the fitness cycle: ECG recording unit location.

signal $x(t)$) by the following integral

$$X(\tau, \omega) = \mathbf{STFT}\{x\}(\tau, \omega) = \int_{-\infty}^{+\infty} x(t)w(t - \tau)e^{-j\omega t} dt, \quad (2.6)$$

where $w(t)$ is a window convolving over the complex signal $x(t)e^{-j\omega t}$ [12]. The window must be described by a limited function, which must respect the following normalization condition:

$$\int_{-\infty}^{+\infty} w(t)dt = 1. \quad (2.7)$$

Then, the spectrogram is the square magnitude (usually expressed in dB) of the quantity defined in (2.6), namely:

$$\text{spectrogram}(\tau, \omega) = 10 \log |X(\tau, \omega)|^2. \quad (2.8)$$

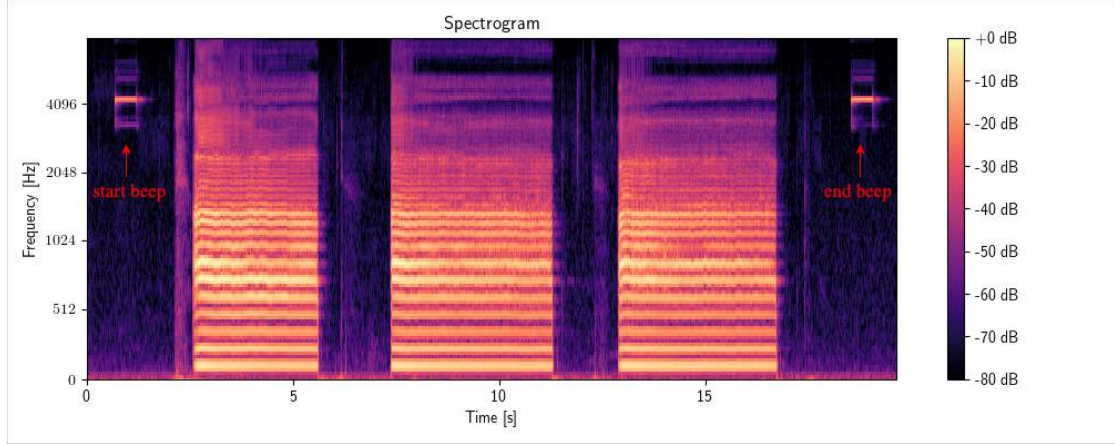
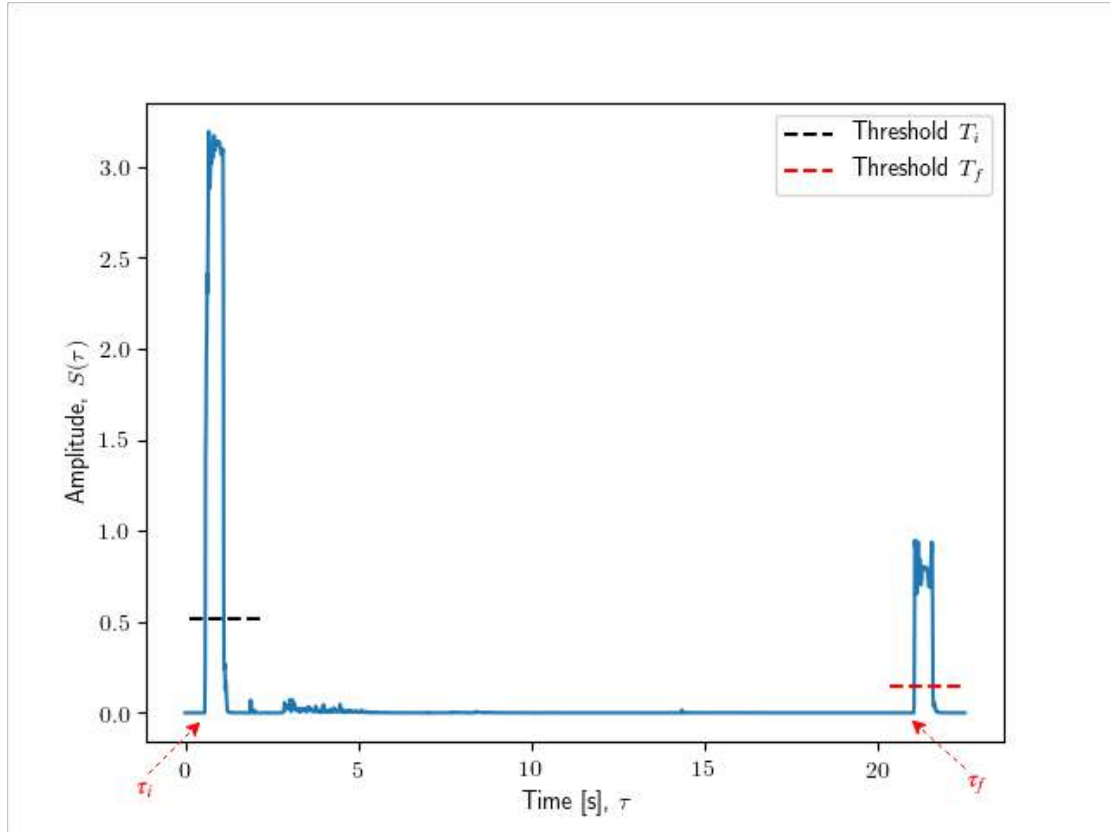


Figure 2.13: Mel spectrogram of three /a/ vowels, computed over the interval $[0, 8000]$ Hz.

The STFT is a Fourier-related transform used to determine the sinusoidal frequency and phase content of local sections of a signal as it changes over time [46]. In practice, the procedure for computing STFTs is to divide a longer time signal into shorter segments of equal length and then compute the FFT separately on each shorter segment. This reveals the frequency spectrum on each shorter segment. Then, one usually plots the changing spectra as a function of time, obtaining a spectrogram over a bidimensional grid depending on time and frequency, with the magnitude of the spectra expressed on a graduation of colors. Understanding this, a pure sinusoidal segment of a known frequency f_0 will be represented as a horizontal segment in the spectrogram. For this study case, the start and end beeps are produced with a known frequency ($f_0 = 4400$ Hz, corresponding to $C^\#8$ tone), thus they will be represented in the spectrogram in a specific location. However, they will not appear as two segment lines because the acoustic beeps are not pure sinusoidal tones, but rather they will occupy a region of the spectrogram (Figure 2.13). Therefore, to find τ_i , τ_f , it is sufficient to extract

$$S(\tau) = \text{spectrogram}(\tau, \omega) \Big|_{\omega=\omega_0}, \quad (2.9)$$

which is a horizontal line at $\omega_0 = 2\pi \cdot 4400$ rad/s. On that signal, it is easy to identify those values, as shown in Figure 2.14. In fact, it will suffice to define two signal thresholds, T_i for the start beep and T_f for the end beep, and then apply them for an onset detection of τ_i and τ_f on

Figure 2.14: Spectrogram line from Figure 2.13 at $f_0 = 4400$ Hz.

the signal $S(\tau)$. In this view, τ_i and τ_f can be expressed formally as

$$\begin{cases} \tau_i = \inf\{\tau \in I_i : S(\tau) \geq T_i\}, & (2.10a) \\ \tau_f = \inf\{\tau \in I_f : S(\tau) \geq T_f\}. & (2.10b) \end{cases}$$

For the implementation of the intervals and the threshold, they have been chosen of equal duration, peer to 3 s, since the audio beeps are produced relatively near the start and the end of the recording. This choice implies an audio recording duration by default longer then 6 s. Hence, the thresholds T_i and T_f were defined as the mean value of $S(\tau)$ over these intervals, namely

$$\begin{cases} T_i = \text{mean}\{S(\tau)\} & \tau \in I_i, & (2.11a) \\ T_f = \text{mean}\{S(\tau)\} & \tau \in I_f. & (2.11b) \end{cases}$$

The choice of employing two different thresholds defined over intervals was motivated by the fact that the two peaks of $S(\tau)$ could exhibit different amplitudes, making impractical to define a single threshold for both. Moreover, the fact that the acquisition system inherently executes the `beep()` function after the start of audio recording (and before the end of it), ensures that the beeps are encoded only in the starting and ending intervals of the audio track, making then feasible to use only the first and last three seconds for defining the threshold. At this point, the τ_i and τ_f are identified in the audio signal. Yet, the found values are not precise enough, i.e. they do not mark exactly the rising edges of the start and end beeps. In fact, they have been theoretically defined using the continuous-time STFT; in practice, the discrete STFT of a discrete signal is a finite two-dimensional grid. The resolution of this grid is bound by the Gabor's uncertainty principle, which states that

$$\sigma_t \sigma_f \leq \frac{1}{2}, \quad (2.12)$$

where σ_t , σ_f are the standard deviation of time and frequency for a given sliding window, and can be interpreted as the grid resolutions along the time and frequency axes respectively [21]. That means that a narrow sliding window ($\sigma_t \equiv \Delta t$ is small) will give a spectrogram with a low resolution in frequency ($\sigma_f \equiv \Delta f$ is big), and vice versa for a wide sliding window. The time and frequency resolutions of a discrete STFT are thus influenced by:

- the choice of the window $w[n]$ (Gaussian, Hamming, Bartlett, etc.);
- the length of the window, i.e. the sample length N of $w[n]$;
- the amount of overlap q of the sliding window, i.e. the fraction of samples in common when the window is moved by one step.

Therefore, in general it holds that

$$\Delta t = (1 - q)NT_s, \quad (2.13)$$

and by the (2.12)

$$\Delta f \leq \frac{1}{2} \frac{1}{(1 - q)NT_s} = \frac{B}{(1 - q)N}, \quad (2.14)$$

where $B = f_s/2$ is the Nyquist frequency. Therefore, the values of τ_i and τ_f are determined on

the spectrogram line with a smaller resolution as if they were determined on the signal in its original time-domain representation. Moreover, the values identified by means of (2.10a) and (2.10b), may be imprecise considering that the steepness of the rising edges in Figure 2.14 have their own rising time and do not intercept with the thresholds as vertical lines. Nonetheless, this approach results appropriate to easily identify them because of the well-known time-frequency location of the beeps, which is on the other hand not trivial when examining the signal in time-domain. However, to exactly locate τ_i and τ_f and to overcome the aforementioned issues, a refinement process for both time instants was devised on the hypothesis that the real values of τ_i and τ_f fall in the neighborhood of those found using the spectrogram approach. In fact, as shown in Figure 2.15, the τ_f found with the spectrogram technique (in magenta color) is located farther of about 20 ms from the actual rising edge. After the refinement, the said value is now located (in red color) exactly in correspondence of the said rising edge.

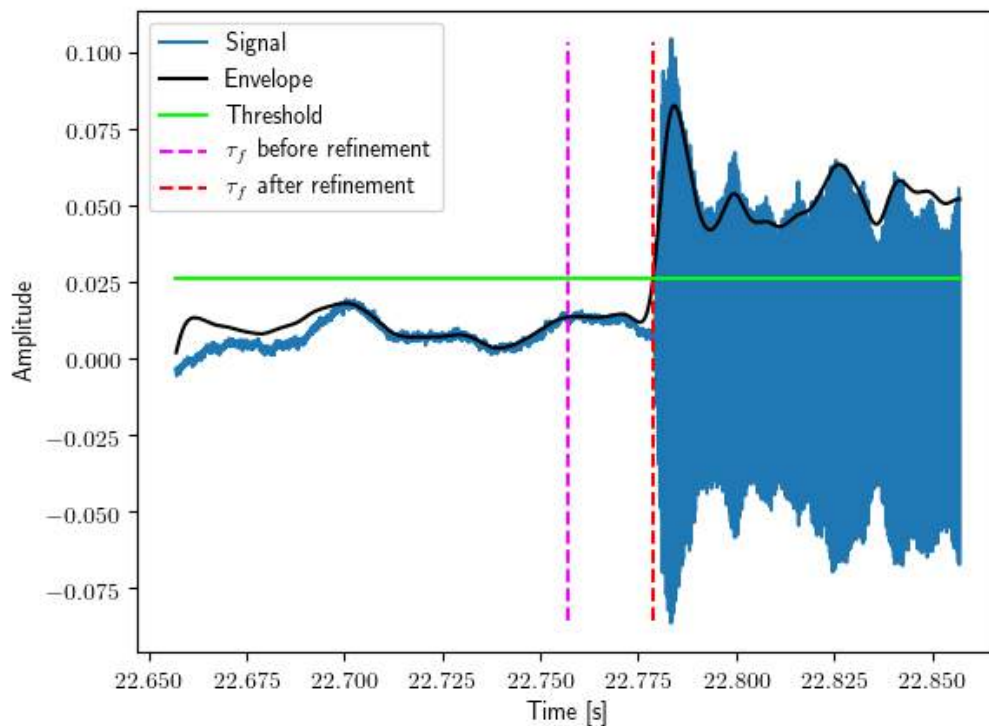


Figure 2.15: Example of refinement of τ_f .

The said refinement is articulated, say for τ_f , in the following steps:

1. take a neighborhood of τ_f , with a radius of 100 ms, obtaining a 200 ms interval of the signal. The obtained interval is bisected by τ_f in two sub-intervals: the one on the left-hand side containing only noise (indicated as I_{noise}) and the other on the right-hand side containing the beep (indicated as I_{signal});
2. calculate a threshold over I_{noise} using the following formula

$$T = \text{mean}\{s(t)\} + h \cdot \text{std}\{s(t)\} \quad t \in I_{\text{noise}}, \quad (2.15)$$

where $s(t)$ is now the time-domain version of the audio signal, h (in this case $h = 2$) is a preset parameter, and the $\text{std}\{\cdot\}$ indicates the standard deviation. This threshold is based on common outlier detection techniques, and it has been employed to detect abrupt temporal changes on signals (like in standard electromyography [3, 13, 56]).

3. define the envelope of the signal $s(t)$ by taking the module of the analytic signal $z(t) = s(t) + j\hat{s}(t)$, where $\hat{s}(t)$ is the Hilbert Transform of $s(t)$ [31];
4. find the intercept between the envelop and the threshold: this value is assigned to be the new value of τ_f .

All the steps so far discussed are summarized in the scheme of (Figure 2.17), and apply exactly also for τ_i detection. The rationale behind this implementation is that before the beep, the signal $s(t)$ is essentially containing noise and therefore it is not overcoming the threshold T . As soon as the beeps are produced, the waveform of $s(t)$ results in tight oscillations and its envelop overcomes the defined threshold in correspondence of the sought instant. Finally, once that the refinement process is completed, τ_i and τ_f are compensated for the encoding delay expressed in (2.5), hence they are both subtracted by δ . To be noted, the algorithm assumes that the interval I_{noise} actually contains neither uttered words nor voice sounds. Conversely, the calculation of the threshold would be biased and the refinement erroneous. To overcome this shortcoming, the environment must be quiet and silent when the beeps occur, and in particular the subject should not speak or produce any noise at the start and at the end of the acquisition.

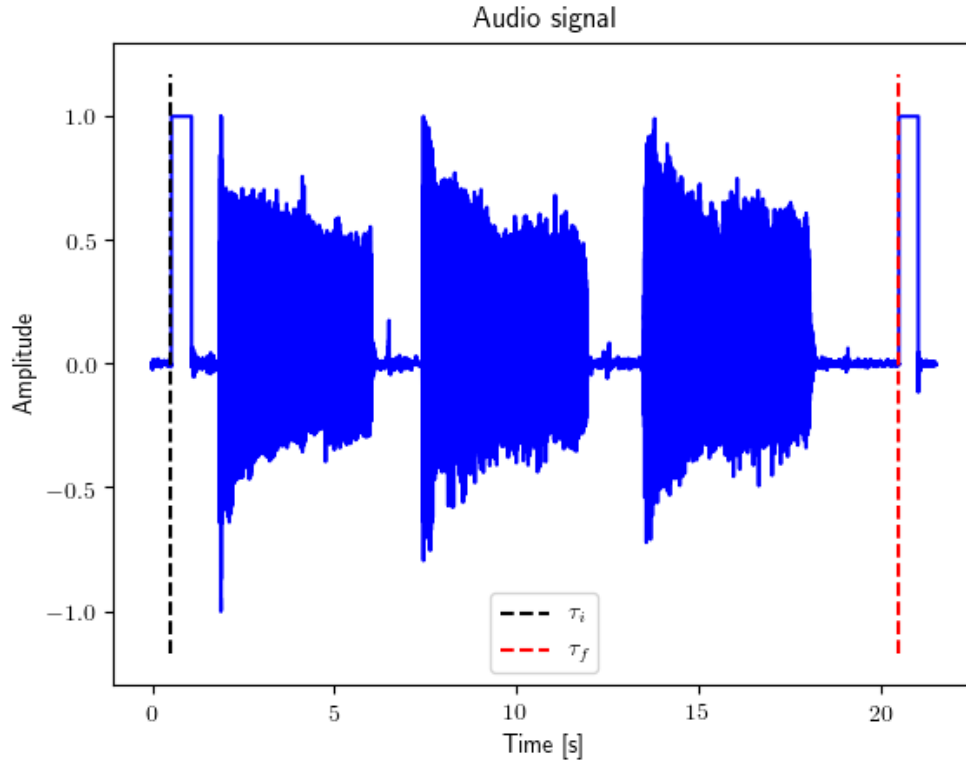


Figure 2.16: Original waveform giving the spectrogram of Figure 2.13, now with rectangular impulses replacing beeps.

Once that the rising edges of both the start and end beeps are found, also their falling edges are automatically identified by knowing the duration of one beep ($\Delta t_{\text{beep}} = 0.5$ s). The beeping intervals are thereby located and the signal segments associated with them are substituted with ones, allowing to mark those clearly and make them intelligible from both visual inspection of the signal waveform (see fig. 2.16) and audio reproduction.

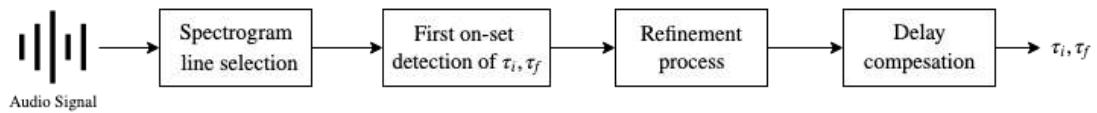


Figure 2.17: Scheme of the beep detection algorithm.

Chapter 3

The BioMiner Application

As discussed in the previous chapter, the acquisition system requires several processes to be executed by the computer and usually in *time-sharing* fashion. Therefore, the discussion here focuses on the software implementation and architecture of the BioMinerTM application, developed for audEERING GmbH to collect and label audio/ECG data in different possible situations and experiments. The persons entitled to use the application must be involved directly or indirectly with audEERING GmbH, which is also the seat where the application is used. This application is therefore for private use, and one or more persons in possess of the above requirement are usually demanded also to organize the data collection sessions, and to explain and make aware participants about the contents and the conditions of the experiments before any recording or acquisition begins. The application was devised on the basis of the following principles:

- *easiness of use*, i.e. the correct use of the acquisition system in a simple and intuitive way. The software components automatically execute the programs needed to collect audio and ECG at the same time without any further necessary action;
- *customizability*, to set the desired features and appearance of the application during a recording session; for example, tasks per session, duration of a single trial, number of acquisitions needed to complete a session, and so on;

- *control*, to check and monitor the execution flow of the ongoing acquisition. This means to be able to start, stop, save, and start again a recording session, to have a clue of the quality of the ECG signal being collected, and to eventually interrupt the acquisition if the signal is highly noisy or corrupted;
- *interactivity*, i.e. the participant can use the application to scroll among different types of experiments, to have a visual feedback from the screen of the ECG being acquired, to regulate the amount of exercise by inspecting the BPMs displayed on the screen;
- *storage*, i.e. the capability of software components to store, annotate and label data at the end of a recording session in a meaningful and orderly scheme.

All these principles brought to the development of a computer application written in Python, provided with a Graphical User Interface (GUI) developed using PyQt (version 5.9.2), one of the most popular Python bindings for the Qt cross-platform C++ framework. The application runs on distributions with Linux operating systems. It was conceived to record spoken utterances of different kind limited to close-room environments, provided with the add of one physiological signal, simultaneously collected. In this case, the experimental setup aimed to collect and store audio/ECG data, and analyze and label them in post-acquisition. Nevertheless, this application is provided with specific re-programmable components, allowing to use it with different setups.

3.1 Features and Modalities

The application is made of a set of different panels displaying different features, depending on the modality being used. There are two possible operative modalities: the **acquisition mode**, which is used to acquire data, and the **analysis mode**, which allows to load and inspect back again records from the dataset. Henceforth the following terms will be used without ambiguity, according to these definitions:

- **dataset**, referring to the digital set of organized audio/ECG data;
- **participant**, referring to the person participating to data collection and interacting with the application using the acquisition mode, as opposed to **user**, referring to who instructs the participant during the acquisition and uses the application in all its modalities.

- **session** (or **record**), referring to a sequence of trials that the participant has to fulfill to complete one;
- **trial** (or **experiment**), referring to a participant's acquisition made between start and end beeps;
- **task**, referring to the set of actions (e.g. producing several sustained vowels in row) that the participant has to carry out along the trial duration;
- **subtask**, referring to a single action made following application's instructions.

3.1.1 Acquisition Mode

The acquisition mode is the default modality. As shown in Figure 3.1, when the application is launched, the main window is divided into three parts: the *Scope* window on the left-hand side, the *I/O panel* on the right-hand side, and the *Options* box in the bottom. There is also a *BPM* box to show the current BPM for an ongoing trail.

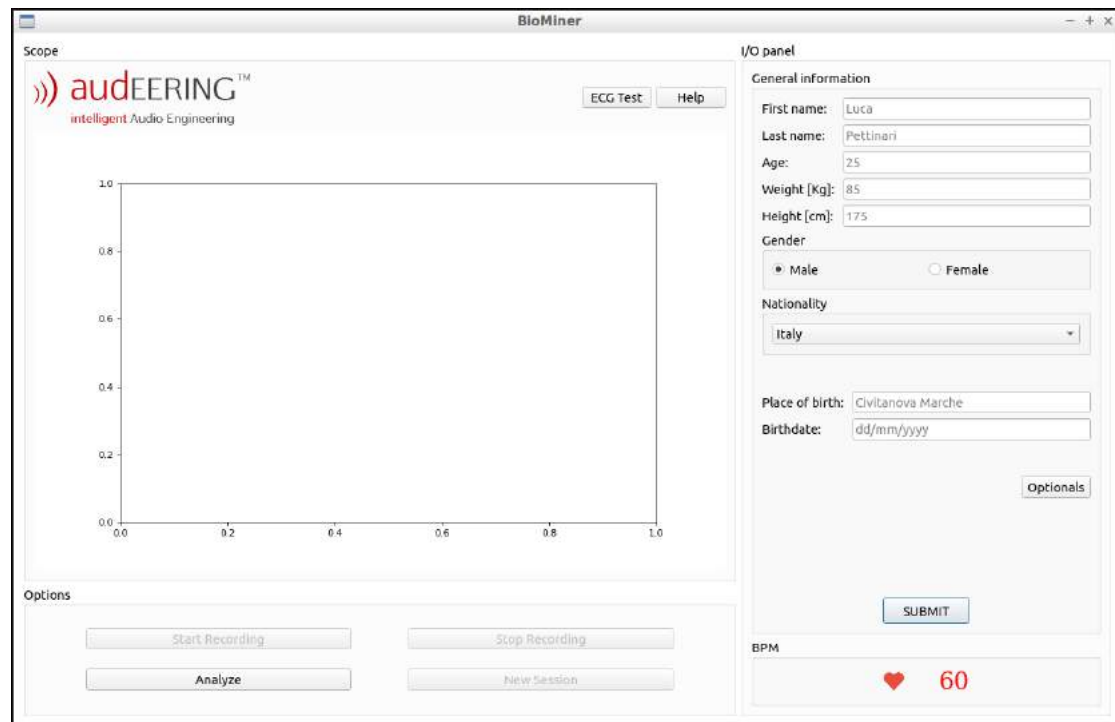
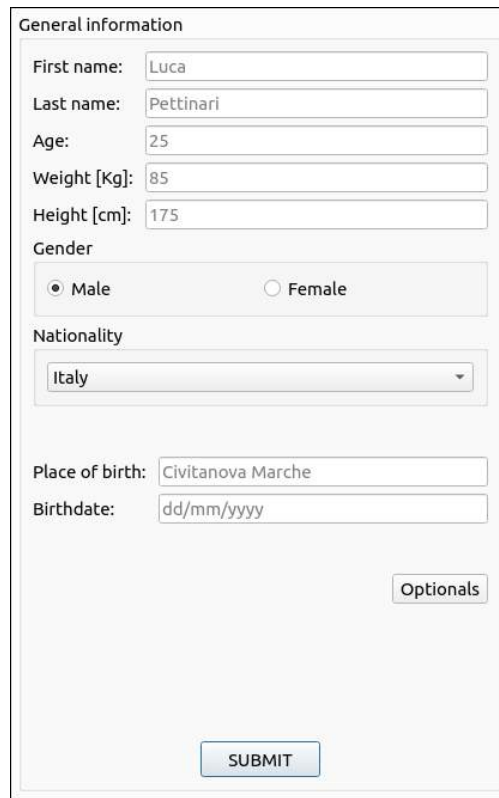


Figure 3.1: Initial screen.



The form is titled "General information" and contains the following fields and controls:

- First name:** Text input field with the value "Luca".
- Last name:** Text input field with the value "Pettinari".
- Age:** Text input field with the value "25".
- Weight [Kg]:** Text input field with the value "85".
- Height [cm]:** Text input field with the value "175".
- Gender:** Radio button group with "Male" selected and "Female" unselected.
- Nationality:** Dropdown menu with "Italy" selected.
- Place of birth:** Text input field with the value "Civitanova Marche".
- Birthdate:** Text input field with the placeholder "dd/mm/yyyy".
- Optional:** A button labeled "Optional" located to the right of the birthdate field.
- SUBMIT:** A button at the bottom center of the form.

Figure 3.2: *General information* box.

To begin with the recording, the participant must first provide general information by fulfilling the *General information* box (Figure 3.2). To this one corresponds the following entries: *First name*, *Last name*, *Age*, *Weight*, *Height*, *Gender*, *Nationality*, *Place of birth* and *Birthdate*. These are required to have a description of every subject of the dataset, allowing thereby to later evaluate several statistics about it. To be noted, some of the said information is personal, i.e. it could be used to identify the subject related to one or more records in the dataset and must be treated under clear and explicit conditions, complying with ethical and privacy norms. First name, last name, place and date of birth represent a set of information that can directly identify a subject, and thus are not saved permanently within the dataset in any form. They are only required to produce a unique participant code (by means of a one-way proprietary hash function) that will be stored with the other metadata, making thereby any record of the dataset anonymous. Once all the primary information required is correctly filled, the participant can decide to start the recording after pressing the *SUBMIT* button. If no information is submitted,

Health and lifestyle

Lifestyle:

☒ Sedentary ☐ Active

Habits:

☐ Smoking

☒ Alcohol Rare

☒ Caffeine Regular

☐ Drugs/Meds

☒ Stress Moderate

Medical conditions:

☒ Cold or flu

☐ Stuttering

☐ Tooth aches

☐ Swallowing problems

Previous diseases:

List here your precedent illness or diseases...

Figure 3.3: Example of *Health and lifestyle* form filling.

the session will be considered a test and stored in the dataset as such. By pressing the *Options* button instead, the participant accesses a new area to provide secondary information. The I/O panel changes accordingly, swapping the entries and widgets of Figure 3.2 with the ones shown in Figure 3.3. In this part, the participant can specify information about the type of lifestyle led, like the attitude towards sport, tendency to unhealthy habits, or suffering of medical conditions at the moment of the session. In certain cases, the participant can also specify diseases suffered in the past, but now absent or partially resolved. The *Sedentary/Active* radio buttons is a mutually exclusive pair, while the other options can be chosen by ticking the corresponding checkbox. In particular, for the *Habits* section, they are all associated with a correspondent slider to indicate the intensity over five levels: *None*, *Rare*, *Moderate*, *Regular*, and *Intensive*. The default option is "None", and it remains so if the associated check-box rests unticked. It must be stressed out that the part referred to healthy and unhealthy habits holds particular importance. In fact, these



Figure 3.4: Screen of the acquisition mode.

factors influence in various ways heart rate variability, and thus they can be used to provide a better understanding of characteristic trends appearing in the collected ECGs [4, 45]. As well for voice, among the available medical conditions, there are only those affecting the capability of speech and phonation. Once completed, the participant can finally press the *SUBMIT* button and proceed with the first task of the session by pressing the *Start Recording* button. At this point of the session, all the information provided is saved in a *metadata buffer*, a Python dictionary containing general participant information and task annotations. This one will be later saved and stored as a metadata *.json* file at the end of the session. In Figure 3.4, it can be seen how the acquisition mode works after submitting. The *Scope* window displays in real-time the ECG signal being shown over a fixed window of observation (sliding one second every second), with the BPM information calculated on it. By means of the *Option* box, the user can stop the trial any time and even decide to unsave it, if something goes wrong. The *I/O panel* is used to have control over the task to be performed. The upper side contains a title, indicating the category of the task to be done. Then, the picture below refers to the task or eventually to one

of the subtasks making up a single task. For example, in the case reported in Figure 3.4, the title "Vowels" refers in general to the category of the task at hand, while the underlying picture pertains to one of the subtasks (which in this case are five, each one for every vowel). In this way, the participant is showed exactly what to do during a task, without the intervention of the user. Moreover, the participant has control over the ongoing task. In fact, considering again the example of Figure 3.4, by means of the *Subtask navbar* box, the user can autonomously decide which subtask to do and in which order, and eventually skip one, if not comfortable with it.

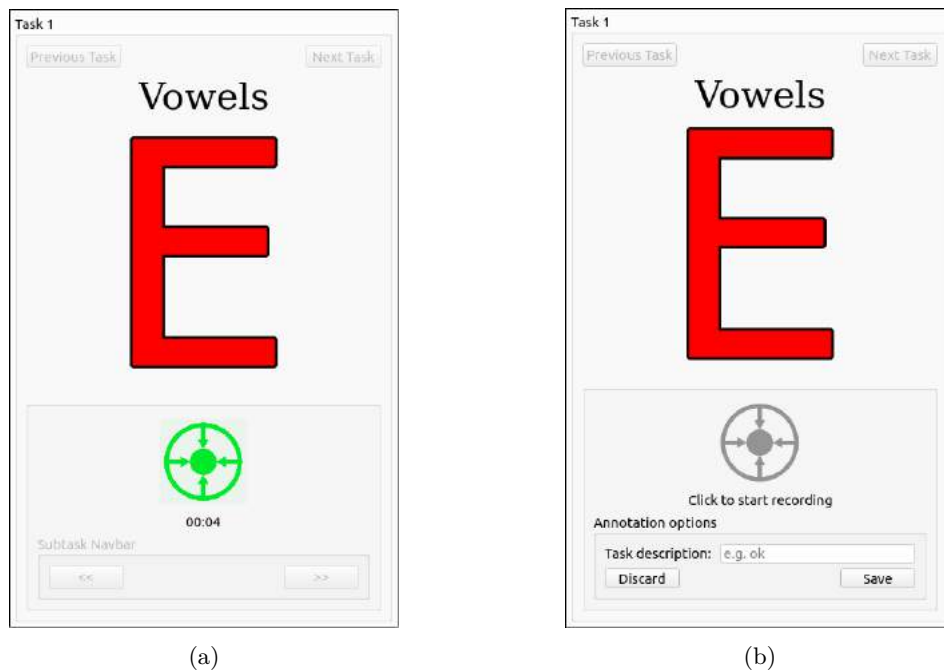


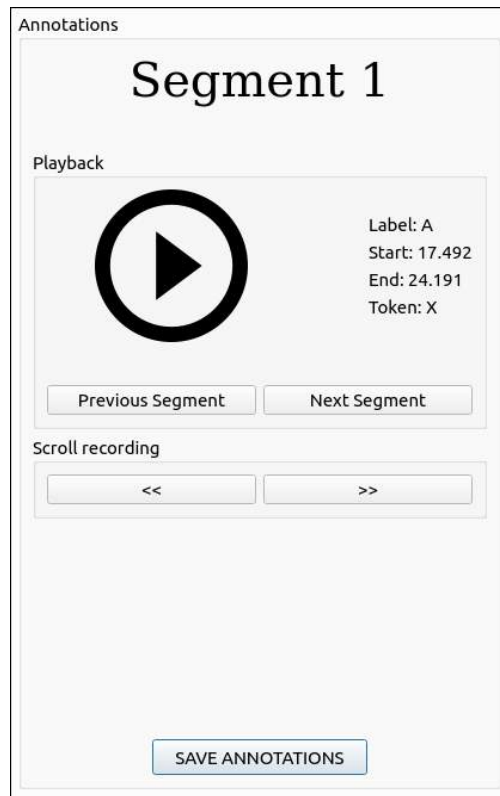
Figure 3.5: Fulfillment of an /e/ sustained vowel with a timer (Figure 3.5a). End of the same task with description and saving options (Figure 3.5b).

With reference to Figure 3.4, the participant chooses when to start the vowel displayed by pressing the red target button below the picture (see Figures 3.5a, 3.5b). This button will turn from red to green (fig. 3.5a) and start a timer to give the participant the possibility of controlling the length of the subtask duration. Once ended, the button can be pressed again, and the participant is prompted to an optional short description of the completed subtask. Then, the user will choose to save or discard it. For the sake of clarity, the *Discard* option does not mean that either the audio or the ECG segments related to a given subtask are canceled, but rather

it means that the said segment is annotated as a "discarded" one, i.e. as something which is not consistent with the description of the subtask itself. For instance, if a participant records a too short or too long sustained vowel, or a sustained vowel with multiple interruptions, the related audio segment is regarded as a wrong take and therefore useless compared to other segments properly recorded. Therefore, the target button is of importance for the *segmentation* of the audio track associated with every record. In fact, if the final goal is to have a set of instances representing the same situation (e.g. a set of sustained vowels), then it is extremely important to know the timestamps at which the corresponding events took place during the trial. The pressing of the target button saves segment's start and end timestamps, its label, and a short description (if given) of the segment. This data is temporally kept in the metadata buffer, and later saved in the metadata file permanently. When a series of task is completed, the participant can press the *New Session* button: the saving options will allow to store all the recordings and metadata associated to tasks and subtasks as a structured record of the dataset, and a new initial screen (like the one in Figure 3.1) will be prompted to the participant. It must be noted that the succession of tasks constituting a trial is precisely defined in the **experimental protocol** (see §5.1.1), where it is specified the number of tasks per session, and the number of subtasks per task. If a task contains a single subtask, these two concepts obviously coincides and there is not the necessity of the *Subtask navbar* panel. For example, in this study, a subtask-free kind of trial is text reading: this is considered as a task on its own, and the entire audio track can be thought of as a container of a single, long event, for which no segmentation, and thus annotation, is done. When a given task is completed by the participant, the corresponding trial can be either stopped by the *Stop Recording* button (saving options will be prompted) or by clicking on *Next Task* button (upper right corner) which will automatically save and move to the session's next task.

3.1.2 Analysis Mode

The analysis mode can be chosen from the initial screen by pressing the *Analyse* button. This will immediately open a file browser which will allow to navigate through the file system, in particular to the path where the desired record to analyze is located.

Figure 3.6: *Annotations* box.

With this feature, it is possible to load only one trial by one. Therefore, the file browser will actually only load from nodes at *trial level* on the directory tree (see §5.1.2). Once the loading is completed, the screen would appear akin to Figure 3.7. In this one, the *Scope* window has widened to the bottom margin, showing now both audio and ECG signals. As for the acquisition mode, the *I/O panel* is used to have control on the data loaded, which eventually contains annotated segments. As shown in Figure 3.6, this panel can either scroll directly over segments (*Previous Segment* and *Next Segment* buttons) or just move the window of observation by one second long step either forwards or backwards. Under the title indicating the current segment, a playback button allows to listen again to the segment. On its right-hand side, there is also a description of the segment. For instance, in the example of Figure 3.6, the *Label* field indicates that an /a/ vowel corresponds to this segment, with start and end timestamps respectively reported in *Start* and *End* fields. The *Token* field assumes only "V" or "X" values, respectively meaning a saved or discarded segment. The *Scope* window adjusts to show the

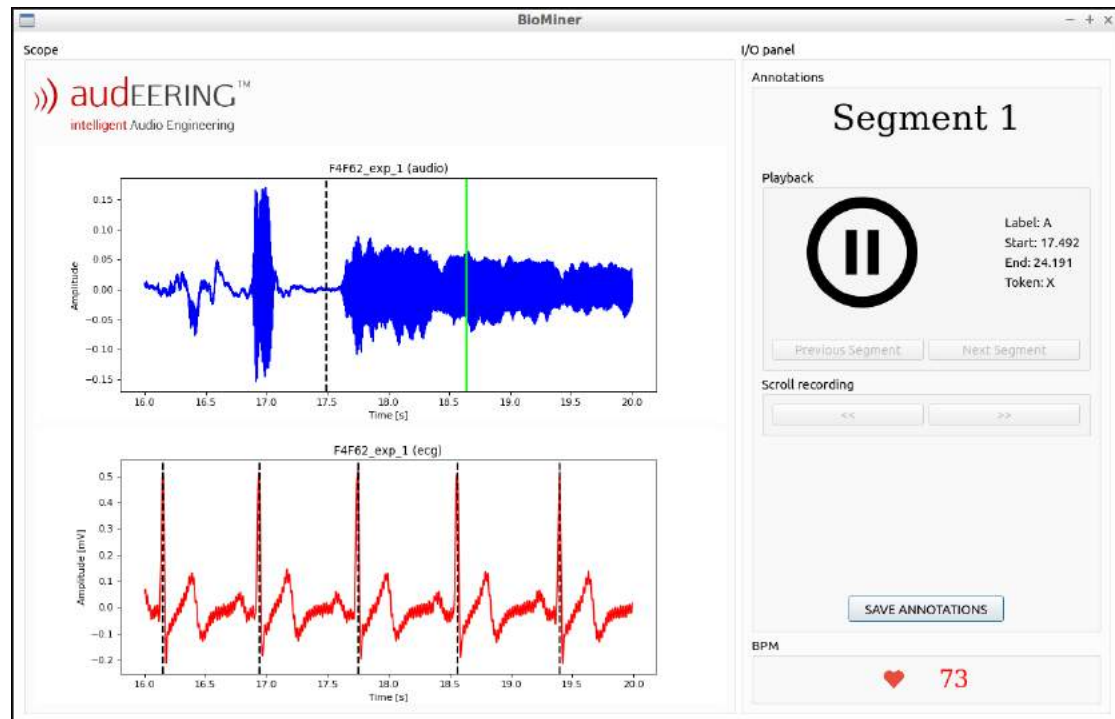


Figure 3.7: Screen of the analysis mode.

current segment, with start and end timestamps marked with black dashed vertical lines. These lines are click-sensitive and can be freely moved by the user. For example, if the overall content of a segment complies with the subtask but the final part contains a short noisy sound coming from a slamming door, the user might decide to exclude that noise anticipating the end of the segment by moving a bit backwards the line. This is why the playback feature is important: albeit every segment could have been already provided with reliable timestamps, in some cases there might be the necessity to refine the segments to produce more refined ones. Thus, the best results are obtained with a *play and cut* strategy, where the user can listen again to the audio associated to the current segment, cut it and continue this process until the result is satisfactory. The same holds for the ECG track, shown below the audio signal. In fact, as it will become clear in §3.3, the computation of the heartbeat highly depends on how well the peaks are detected in the signal. As shown in Figure 3.7, these peaks are pinpointed by dashed black lines according to the heartbeat detection algorithm. Although its high detection accuracy (see §5.1.2), the algorithm might produce erroneous peak locations (especially in noisy conditions)

which can be corrected manually, moving the lines until they overlap with the heartbeat spike: this new line location (in time) will substitute the former, erroneous one. In addition, marker lines can be deleted should the algorithm recognize as heartbeat a spurious peak in the signal, likely produced by noise, electrode misplacement or movement artifacts. Such lines can be added as well if one or more beats are missed. When the analysis (i.e. scrolling and eventual correction of the synchronized audio/ECG track associated to a trial) is finally completed, the user can update the metadata file related to the trial with new annotations (e.g. refined timestamps for a given segment) by pressing the *SAVE ANNOTATION* button. It is important to stress out that these two capabilities of the application, i.e. to inspect and manually correct both ECG and audio tracks, are together extremely important to help the user with the so called dataset **sanity-check**, a customary practice in data collection that is always abode by machine learning practitioners, and most of the time realized with usually laborious and time-consuming annotation work.

3.2 Software Architecture

After having described the features of the application and its uses, here it is presented its software architecture, depicted in Figure 3.8 using a UML-like representation. The application runs on a Python environment (the installation of Python3 or higher is required) by instantiating an object from the `BioMinerUserInterface` class (Figure 3.8, yellow block) and calling its `exec_()` method, which makes execute the application's main loop. This class generates all the interactive elements of the GUI, like layers, buttons, canvas windows, sliders, input slots, etc. All these graphic elements are objects defined by standard PyQt libraries, and are called *widgets* (Figure 3.8, grey box). These widgets are usually linked with *callbacks*, i.e. member functions triggered on the emission of a signal detected by the widget itself. For example, when a button is clicked, it emits a *clicked* signal and the instructions contained in the associated callback can be executed. In addition to the `BioMinerUserInterface` class, which allows user interaction during the recording session through the GUI, the `BioMinerSession` class (Figure 3.8, pale-blue block) provides all the I/O components for the actual acquisition and storage of audio/ECG data. As shown in the scheme of Figure 3.8, between these two main classes there is an opposite

arrow couple, suggesting that they interact, mutually using functions and variables each with the other for their own processes. For example, at the end of every trial, `BioMinerUserInterface` conveys all trial annotations present in the metadata buffer to `BioMinerSession`, which will use them to generate a metadata file containing them (through `writeMetadata()`). On the other hand, when `loadTrial()` is called to load a record from the dataset, data must be passed also to `BioMinerUserInterface`, and opportunely processed to be visualized on the windows of the *Scope* panel. In addition to these two main classes, there are two additional ones: the `BeatDetector` class, which implements the algorithm to calculate the heart rate in term of BPM, and the `BeepDetector` class, which implements the beep detection algorithm described in §2.3.1. The first one is invoked exclusively by `BioMinerUserInterface` to produce BPM reads in both acquisition and analysis modalities, whereas the second one is called by `BioMinerSignal` to retrieve the τ_i and τ_f before the record is finally stored in the dataset. To complete the picture, there are also three files needed to configure the application: a configuration file (`conf.json`) to define appearance and uses of the application, a requirements file (`biominerenv.yml`) to load all the libraries necessary to the Python interpreter to correctly execute the application, and a Python script (`lookup_participant_hash.py`) which implements the already mentioned one-way hash function to render anonymously records of the dataset.

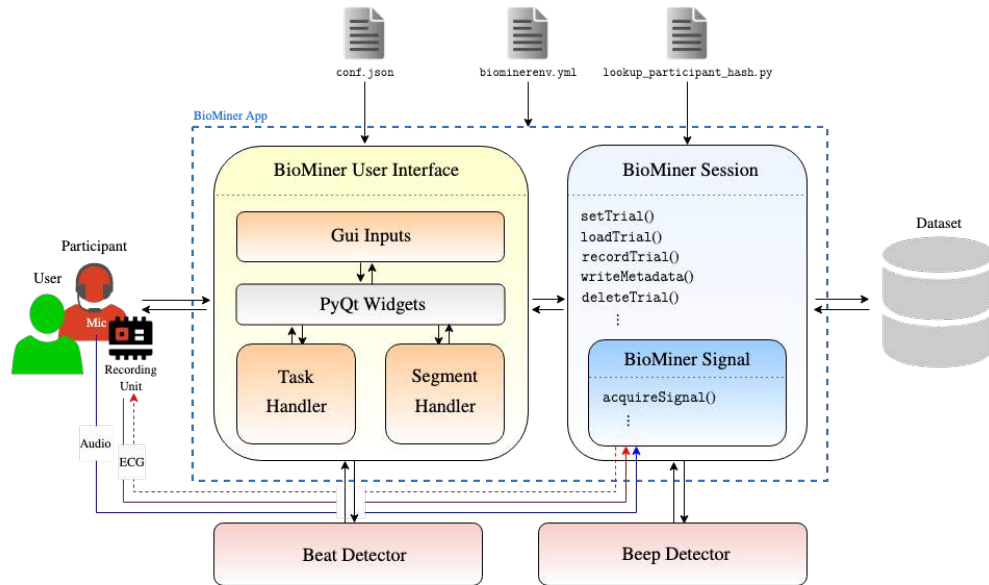


Figure 3.8: Class diagram of the application.

3.2.1 The Configuration File

The configuration file `conf.json` allows to use the application in different experimental setups and it is written to define the appearance and, most importantly, the succession of tasks and subtasks for recording sessions, according to an experimental protocol. An example of a configuration file is given in the following below.

```
{
  "Options": {
    "age": [1, 99],
    "weight": [1, 150],
    "height": [100, 200],
    "duration": [6, 3600],
    "rest_bpm": 60,
    "taper_window": 4
  },

  "Tasks": [

    {
      "task_title": "Vowels",
      "subtasks": [
        {
          "label": "A",
          "picture_path": "figures/task_figures/vowels/A.png",
          "subtask_title": null
        },
        {
          "label": "E",
          "picture_path": "figures/task_figures/vowels/E.png",
          "subtask_title": null
        },
        {
          "label": "I",
          "picture_path": "figures/task_figures/vowels/I.png",
          "subtask_title": null
        }
      ]
    }
  ]
}
```

```

    ],
    "target_button": true,
    "bpm_threshold": null,
    "folder_name": null,
    "random": true
  },

  {
    "task_title": "Reading",
    "subtasks": [
      {
        "label": "reading",
        "picture_path": "figures/task_figures/reading/reading.png",
        "subtask_title": null
      }
    ],
    "target_button": false,
    "bpm_threshold": null,
    "folder_name": null,
    "random": null
  }
]
}

```

The JSON object defined in this file has two main *attributes*: **Options** and **Tasks**. In the first one, admissible ranges for the input fields present in Figure 3.2 are defined; moreover, the minimum (6 s) and maximum (1 h) duration intervals admitted for a session are expressed as well as the initial BPM value (60) shown in the initial screen (as in Figure 3.1) and length of the window of the *Scope* panel in acquisition mode (4 s). The second one encodes the experimental protocol, where its **Tasks** attribute is associated to a list containing in turn a succession of objects (in JSON they are always delimited by a pair of `{}`, as opposed to lists delimited by `[]`), each of them including all the information related to a single task. The succession of tasks is defined by the order of these task objects in the aforementioned list. As of this case, this list includes only two

elements, hence the participant will be presented with these two tasks, whose order is fixed and specified. Every task object has in turn attributes to specify subtasks and their properties. The most important is the `subtasks` attribute which defines again a list whose elements represent now the subtasks in which a task is subdivided. Other fields are: `task_title`, which defines the title of the current task displayed in the *I/O panel* in acquisition mode (see Figure 3.4); `target_button`, a boolean attribute defining the showing (or not) of the target button in the *I/O panel*; `bpm_threshold`, whose value defines a threshold on the BPMs that, once overcome, interrupts the trial (in this case no given threshold is defined because the field is set to `null`); `folder_name`, which contains a specified path to store data in a location external to the dataset; and `random`, a boolean attribute which, if set to `true`, will shuffle the order of subtasks of the current task, and show them accordingly in the application; if this value is `null`, the subtasks will be shown in plain order, following the list contained in `subtasks` attribute. About this latter one, it is important to describe also the attributes of its single list elements (i.e. *subtask objects*), which are: `label`, the string associated to the audio segment that will be recorded in relationship to the current subtask (e.g. for the first subtask object of the task number one of the example code, the `label` attribute is "A", meaning that every segment related to it will be assigned with this string), `picture_path`, which specifies the relative path (referred to where `conf.json` is located in the filesystem) of the picture shown in the *I/O panel* used in acquisition mode (Figures 3.5a, 3.5b) and `subtask_title`, which contains the title to be shown, above the subtask pictures. About this last one, if this attribute is set to `null`, the title shown will be the one indicated in `task_title`. To be noted, if a task has not a succession of subtasks, then the `subtasks` attribute contains only one subtask object, implying that in such cases the concepts of task and subtask coincide.

3.2.2 The BioMinerUserInterface Class

As already mentioned, this class creates the GUI and all the callbacks associated with elaboration processes coming from external classes. Therefore, every widget is associated to a callback, usually linked to a member functions among the following classes: `BioMinerUserInterface` class itself, internal subclasses (those in orange in Figure 3.8), external subclasses (e.g. `BeatDetector`

class) and `BioMinerSignal` class. Some of these callback are based on *threads* (the process is called inside the `run()` method of a `QThread` object), allowing multiple execution of processes without freezing down the application. For example, the *Start Recording* button is linked to a thread which in turn calls the `recordTrial()` method of `BioMinerSession` inside its `run()` method, avoiding thereby the GUI to become no longer responsive, since the thread launches it as a background process. Other than PyQt Widgets, the `BioMinerUserInterface` class is constituted by three main subclasses:

- the `GuiInputs` class, appointed with the task of parsing all the inputs provided by the participant through the *I/O panel* (Figure 3.2) and checking if they are *semantically correct*. For example, the field *Weight* accepts only digits and point, to express quantities in double floating-point format. Any other input character will be automatically rejected by the check methods of this class, which will prompt error box messages suggesting the type of input consented;
- the `TaskHandler` class, devoted to parse all the instances listed in the configuration file and to set the GUI appearance accordingly. In fact, the application uses the *I/O panel* to show to the participant a succession of tasks and subtasks to fulfill (Figures 3.5a, 3.5b), allowing annotation of timestamps and short description of the segment just completed. The succession of subtasks for a given task (with all related information), the showing of the subtask pictures within the *I/O panel* (Figures 3.5a, 3.5b), the title displayed above it and so forth, are altogether managed by this class, which also takes care of saving the said segment annotations to the metadata buffer, which will be later transcribed then in the metadata file;
- the `SegmentHandler` class, which plays a similar role of `Task Handler`, but in the analysis mode of the application. In fact, the class allows to retrieve segments (if any) contained in the audio track of the loaded node, and allows to display its timestamps markers on the *Scope* window. In addition, it implements all the processing related to the already described play and cut strategy (see §3.1.2), taking trace of the modifications on the segment in the metadata buffer, and finally saving them at the end of the analysis phase.

3.2.3 The BioMinerSession Class

This class represents the core of elaboration and processing of the application, as opposed to the previous classes whose features are way more relatable to the management of I/O capabilities of the application. It furnishes member functions to set, load, acquire, save and delete data. These features are indeed made possible by the following methods:

- **setTrial()**. Given a set of inputs, usually those produced through **GuiInputs**, this method allows to create a node (as an empty folder) at session level, according to the dataset organization (explained in §5.1). This method is called when the *SUBMIT* button of the GUI is pressed;
- **loadTrial()**. This method loads a trial from dataset's nodes at session level, with appropriate instructions to read the files (*.wav*, *.csv*, and *.json*) present in the node. It is called when the analyze mode is chosen by pressing the *Analyze* button;
- **recordTrial()**. This method implements all necessary instructions to use the hardware resources, appropriately following communication protocols (see §2.2) and using low-level libraries (i.e. ALSA for audio recording); it can be used only after **setTrial()** has been executed, and creates the folder of the current trial and its related audio and ECG files inside. It is called in the acquisition mode by pressing either *Start recording* button in the *Option* box or *Next Task* button from the *I/O panel*. It runs on a **while** loop controlled by a trial duration time, which can be also set peer to infinity. In this condition, this method can detect *SIGTERM* signals to stop the acquisition, which are emitted by pressing either *Stop Recording* or *Next Task* buttons.
- **writeMetadata()**. This method takes as input the metadata buffer and converts it at the end of a trial in a list of annotations regarding the participant and the trial itself (like segments labels and timestamps). The obtained file is stored inside the correspondent dataset's node at trial level.
- **deleteTrial()**. It is used to remove permanently a node from the dataset, and it is called through the GUI saving options at the end of a trial (it will remove only the node referred to a precise trial) or at the end of a session (it will remove the whole record).

3.3 Heartbeat Detection Algorithm

In the past sections, all the main classes were presented and described, exception made for the `BeepDetector` and `BeatDetector` classes (Figure 3.8, pink blocks). As already mentioned, the first one implements all the methods described in §2.3.1, whereas the second one implements a heartbeat detection algorithm, which is fundamental for mainly two reasons: to control the intensity of physical exercise during different phases of a recording session, and to provide with reliable BPM labels, for appropriate segment annotation.

3.3.1 Electrocardiogram

The electrocardiogram (ECG) is a crucial diagnostic tool, used to represent the summed electrical activity of all cardiac cells recorded from the surface of the body [27, 51]. In particular, it is given by the measure of a voltage drop between two known points of the body over time. This drop depends on ion currents generated on the thorax surface, which are in turn generated by a time-varying electromagnetic field characterized by propagating wavefronts of depolarization and repolarization throughout the heart, spreading through a complex conduction system and triggered from action potentials of sinoatrial node *pacemaker cells*. Such wavefronts travel in the heart by virtue of the intrinsic excitability of cardiac cells to electrical impulses, whose net spatial action over time is described by precise and well-known waveform morphologies appearing in a healthy patient ECG trace. As shown in Figure 3.9, a usual ECG contains the following components:

- **P wave.** It represents the wave of depolarization that spreads from the sinoatrial node through the atria (*atrial depolarization*). Its normal duration falls between 80 ms to 100 ms;
- **QRS complex.** It corresponds to *ventricular depolarization* and *atrial repolarization* events. However, the first one is way more predominant than the second, hence this waveform is particularly representative of the ventricular depolarization. The QRS complex is made by three distinct waves, with different polarities with respect to the isoelectric line: the **Q wave**, just after the P wave, the **R peak**, i.e. the steep peak corresponding to the swift depolarization of ventricles, and the **S wave**, just after the R peak. Its normal

duration falls between 60 ms to 100 ms;

- **T wave.** It corresponds to the *ventricular repolarization* and last longer than depolarization, and its normal duration is usually not measured.

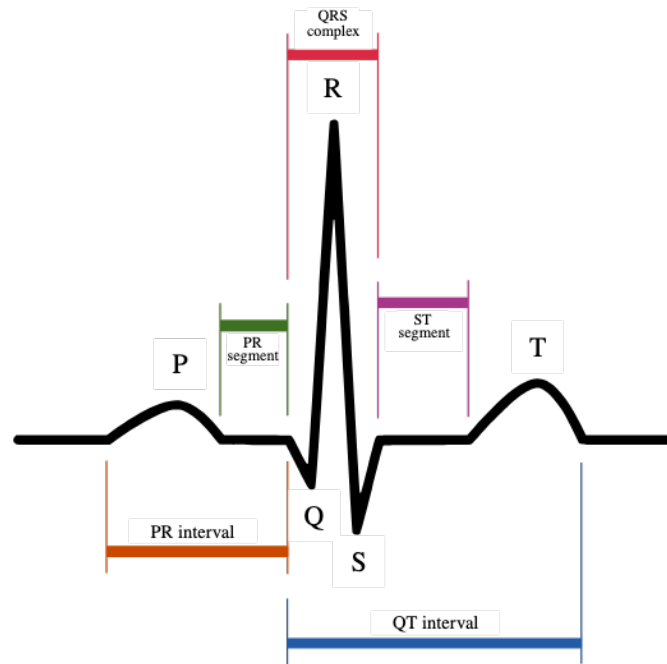


Figure 3.9: Waveform morphologies and segments of a normal ECG tracing (Lead I).

The most important among the described ones is the R wave because it is directly related to the *rapid ejection phase* of the ventricular systole, i.e. the event corresponding to the blood being ejected in the aorta. This is the reason why the identification of consecutive R waves allows to measure the *RR intervals*, whose mean value reciprocal is related to heart rate, which is a fundamental parameter both for clinical diagnoses and engineering applications. An ECG tracing can be derived by measuring a tension drop on the skin surface produced by the heart electrical activity. As shown in Figure 3.10, a measure of such tension on the surface is due to the said ion currents being absorbed by the load (the thorax), and is equal to the voltage difference between the equipotential lines where the correspondent current lines arose from. The heart electrical activity is modeled as an electric dipole. In fact, the global effect of a number of positive and negative charges distributed in a certain region of space, and displacing over time,

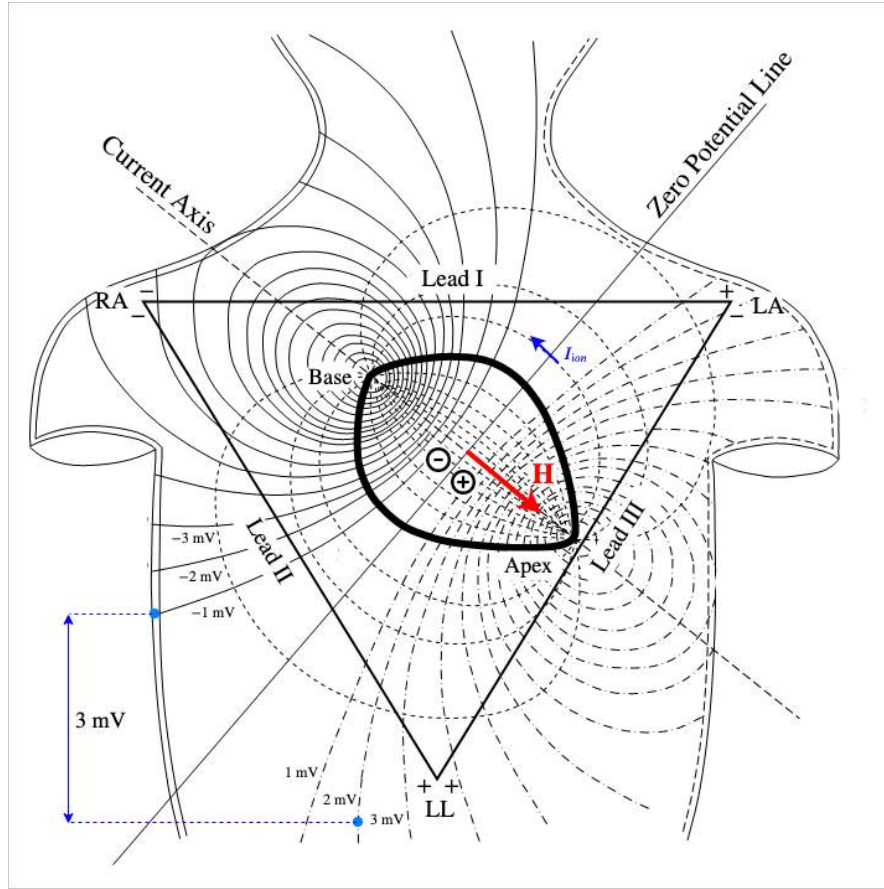


Figure 3.10: The cardiac vector $\mathbf{H}(t)$ and the Einthoven triangle.

can be summarized by two centres of charge, one positive and the other negative, which together create a dipole. The heart can be modeled as a dipole, with the following assumptions:

- centres of charges of equal magnitude q but opposite polarity;
- displacement vector \mathbf{d} with origin on the negative centre and pointing the positive one, with the centre of dipole C placed on the midpoint of \mathbf{d} ;
- \mathbf{d} changes in direction but not in magnitude ($d = \|\mathbf{d}\|$ is called the distance of separation).

With these assumptions, a potential referred to a point P , far from the dipole (e.g. a point on the thorax), can be written at any instant t as

$$V(P, t) = \frac{\boldsymbol{\mu}(t) \cdot \mathbf{r}}{4\pi\epsilon_0\epsilon_r r^3}, \quad (3.1)$$

where $\boldsymbol{\mu} = q\mathbf{d}$ is the dipole moment and $\mathbf{r} = P - C$ is the displacement vector from point C to P, with $r = \|\mathbf{r}\|$ [6]. As appears clear from (3.1), the knowledge of the whole dipole electrical activity is condensed in one single vectorial quantity $\boldsymbol{\mu}$ (instead of vectorial field), which allows to compute every potential at any give point P for any instant t . For this reason, the knowledge of such vector is fundamental for studying the heart electrical activity. Yet, this quantity cannot be directly measured and must be reconstructed from differences of voltage detected on the thorax. To that aim, hereafter it is introduced one of the most fundamental conventions in cardiology: the **Einthoven triangle**, which provides standardized location for electrode positioning and a way to reconstruct the so called **cardiac vector \mathbf{H}** , which is strictly related to the dipole moment. The Einthoven triangle is equilateral and centered in C , so that its vertices – LA , RA and RL following the American Heart Association (AHA) nomenclature – are equidistant from the centre of a quantity R . It is possible to associate to each triangle side a vector with module and direction coinciding with the side itself, and verse pointing from negative to positive polarity. Therefore, to the Einthoven triangle it can be associated a non orthogonal system of vectors $\{\mathbf{a}_I, \mathbf{a}_{II}, \mathbf{a}_{III}\}$. Then, one can consider the difference of potential between two vertices of the triangle. In general, a quantity measured following this convention is called **lead** (or **derivation**) and is expressed – for example for upper side of the triangle – by using (3.1) as

$$\begin{aligned}
 V_I(t) &= \Delta V_{LA,RA} = V_{RA} - V_{LA} = \\
 &= \frac{\boldsymbol{\mu}(t) \cdot (RA - C)}{4\pi\epsilon_0\epsilon_r R^3} - \frac{\boldsymbol{\mu}(t) \cdot (LA - C)}{4\pi\epsilon_0\epsilon_r R^3} = \\
 &= \boldsymbol{\mu}(t) \cdot \frac{(RA - LA)}{4\pi\epsilon_0\epsilon_r R^3} = \frac{\boldsymbol{\mu}(t) \cdot \mathbf{a}_I}{4\pi\epsilon_0\epsilon_r R^3} = \mathbf{H}(t) \cdot \mathbf{a}_I,
 \end{aligned} \tag{3.2}$$

where $\mathbf{a}_I = RA - LA$. The cardiac vector is thus defined as

$$\mathbf{H}(t) := \frac{\boldsymbol{\mu}(t)}{4\pi\epsilon_0\epsilon_r R^3}. \tag{3.3}$$

This quantity has the magnitude of an electric field (V/m). Its intensity and direction vary with the dipole moment, since R is fixed in the Einthoven triangle. Repeating the same calculations

for the other sides, one can obtain the standard ECG leads:

$$\begin{cases} V_I(t) = \mathbf{H}(t) \cdot \mathbf{a}_I, & (3.4a) \\ V_{II}(t) = \mathbf{H}(t) \cdot \mathbf{a}_{II}, & (3.4b) \\ V_{III}(t) = \mathbf{H}(t) \cdot \mathbf{a}_{III}. & (3.4c) \end{cases}$$

Equations (3.4a), (3.4b) and (3.4c) are projection of the cardiac vector along the Einthoven triangle directions [7, 19], and they allow to express it at any instant as

$$\mathbf{H}(t) = V_I(t)\mathbf{a}_I + V_{II}(t)\mathbf{a}_{II} + V_{III}(t)\mathbf{a}_{III}. \quad (3.5)$$

Therefore, the knowledge of voltage differences according to the proposed lead system provide standard ECG traces and allows the reconstruction of the cardiac vector using (3.5). Yet, the tern $\{\mathbf{a}_I, \mathbf{a}_{II}, \mathbf{a}_{III}\}$ does not constitute an orthonormal reference system, and for this reason information about \mathbf{H} is lost and the reconstruction is partial. In fact, the Einthoven triangle theoretically lies on a plane (e.g. the thorax), whereas the \mathbf{H} vector lives in a 3D space. On the other hand, the Einthoven triangle can be seen as a closed electrical mesh, hence the tensions along its sides satisfy the Kirchhoff's first law. In particular, this yields

$$V_{III} - V_{II} + V_I = 0 \implies V_{II} = V_I + V_{III}, \quad (3.6)$$

meaning that only two out of three leads are needed to reconstruct (even partially) the cardiac vector. The complete reconstruction is however possible increasing the number of leads to twelve, and realizing the so called **12-Lead ECG**, which is the standard setup for ECG acquisition in clinical environment. It must be stressed out that the Einthoven triangle gives a standardization for electrode positioning. Nonetheless, in practice the electrodes are never placed to form an exact equilateral triangle, but rather on the upper and lower limbs. In fact, the dipole heart model refers only to the thorax, and the limbs are equipotential zones with respect to the vertices of the Einthoven triangle, thus they are equally suitable for electrode placing as well as the original Einthoven triangle locations. For this reason, LA (left arm) corresponds to the left shoulder or wrist, RA (right arm) to the right shoulder or wrist, and LL (left leg) to the left ankle. An ECG

recording device should thus take at least three inputs, the first two for any given lead and other one for the ground. This latter one is usually collected by one further electrode positioned on the RL (right leg) position.

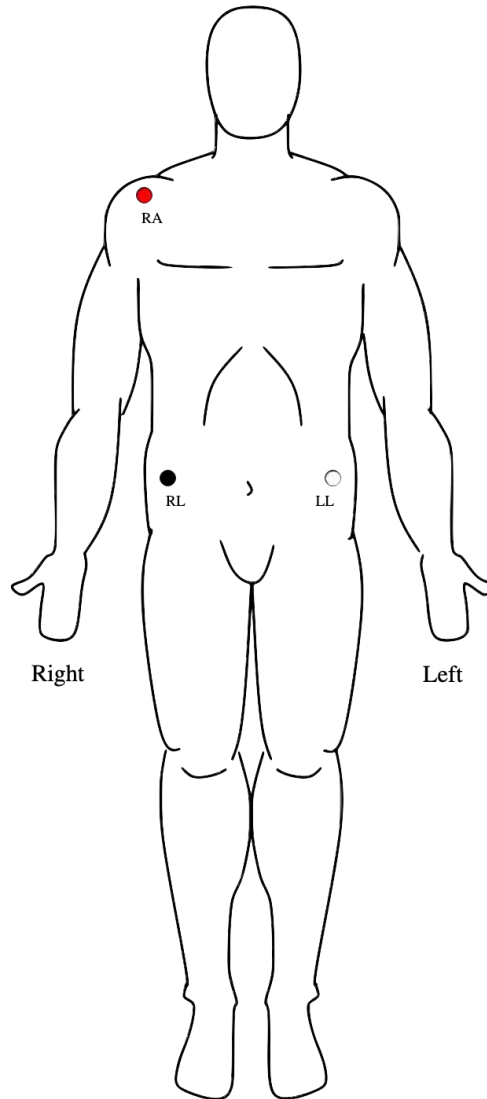


Figure 3.11: Lead II electrode positioning scheme.

In this case study, lead II is used to record ECG from subjects. In fact, as long one is interested only in the study of the heartbeat by detecting the R wave, there is no need of collecting multiple leads to reconstruct the \mathbf{H} vector. Indeed, lead II direction is almost parallel to the projection

of \mathbf{H} on the anatomical frontal plane, making this latter the most used for single-lead ECG studies. As shown in Figure 3.11, the electrode positioning adopted in this study is slightly modified compared to the aforementioned AHA standards. In fact, only RA electrode is placed on the right shoulder, whereas the LL and RL electrodes are shifted from lower limbs to abdomen anatomical landmarks points, corresponding respectively to the left and right anterior superior iliac spines. Nevertheless, this choice does not change dramatically lead II signal characteristics, but indeed reduces movement artifacts and noise levels coming from the activity of shank and foot muscles, especially during exercise periods expected during the recording session.

3.3.2 Heart Rate Measurement

As already mentioned, to measure heart rate from an ECG derivation (e.g. lead II), it is sufficient to find the RR intervals, described by the $RR[k]$ time-series, with $k = 0, \dots, N$, and $N + 1$ counting the number of R peaks detected in the signal. An estimate of the heart rate can be given with the following formula

$$HR = \frac{1}{\mu_{RR}}, \quad (3.7)$$

where μ_{RR} is the mean RR interval. Heart Rate Variability (HRV) is though highly non-stationary, meaning that the statistic descriptors of $RR[k]$ (including μ_{RR}) changes over time [22]. Therefore, the estimate given in (3.7) is not appropriate to describe the fluctuations of heart rate due to HRV. For this reason, other measures have been made available to better follow such fluctuations over time. For example, Beats Per Minute (BPM) is a measure of heart rate counting the number of R peaks occurred in a minute. The shortcoming is that one entire minute of signal is needed before providing a single BPM, making thereby unpractical to provide heart rate continuously. To overcome this, as shown in Figure 3.12, a sliding overlapping window, smaller than 1 min ($\Delta t_{\text{window}} = 4$ s), is employed to select a small portion of the signal and to provide the related BPM, then slides of one second (leaving 3 s of overlap) to produce the successive value, and so on. Then, the first attempt to calculate BPMs for such smaller windows would be

$$BPM[t] = \frac{N_{\text{peaks},t}}{\Delta t_{\text{window}}} \cdot 60, \quad (3.8)$$

where t indicate the current location of the sliding window, $N_{\text{peaks},t}$ is the number of R peaks within it, and 60 is just a scaling factor, since BPM is defined as a measure over a time-length of one minute. Nevertheless, the (3.8) decreases the resolution interval of the computed BPMs. In fact, in this way one would compute only "discrete valued" BPMs, depending on the number of peaks detected within the sliding window (e.g. 60 BPM for four R peaks detected, 75 BPM for five peaks, 90 BPM for six peaks, and so on). Therefore, to solve this issue and regain resolution, BPMs are calculated by finding reciprocal of the mean RR of the current window following the formula (3.9), which resembles the concept expressed in equation (3.7):

$$BPM[t] = \left\lfloor \frac{N_{\text{peaks},t}}{\sum_{k \in J_t} RR[k]} \cdot 60 \right\rfloor, \quad (3.9)$$

where J_t is the set containing the indexes of $RR[k]$ selected by the sliding window at its current location and the floor operator provides only integer values. With this adjustment, the range of values extends to any possible integer and represents the best trade-off to track heartbeat variations over time.

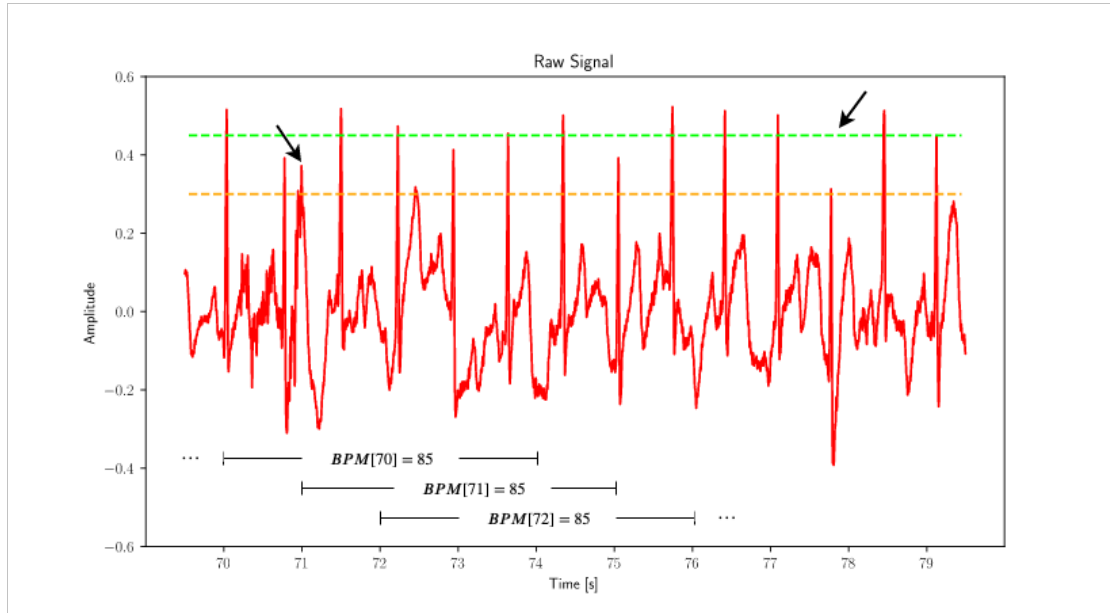


Figure 3.12: Overlapping window method to compute BPMs, with a sliding step of 1 s. Detail: poor detection performance by means of constant thresholds (in orange and green).

As shown in Figure 3.12, a simple constant threshold to discriminate R spikes is performing usually bad, resulting in erroneous R peak detections (a T wave is detected by the threshold coloured in orange in Figure 3.12) or non-detection at all (more than one R peak is not detected by the green threshold, again in Figure 3.12). This is indeed generally refuted also in literature [28, 30], and the widely recognized and accepted approach was firstly described in a paper by Pan and Tompkins [38]. This same approach has been applied to this study, with the slight modification of applying the algorithm to a succession of overlapping segments provided by a sliding window. Every segment is processed in two main steps of processing: signal manipulation with a series of transformations and determination of an adaptive threshold for R peak counting.

3.3.3 Preprocessing Stage

In first place, the input segment selected by the sliding window must be resampled. In fact, as explained in §2.2, the ECG recording device provides non-uniformly sampled signals, making them difficult to treat with standard frequency filtering techniques, which usually requires a sampling frequency to determine the cut-off frequencies of the filter. To overcome this, the ECG signal is interpolated with a cubic spline, upsampling the segment to a sampling frequency of $f_s = 1$ kHz. Then, the obtained signal is subtracted with its own mean value to obtain a zero-mean signal. At this point, the segment is processed as follow:

- **removal of P and T waves.** P waves and T waves are removed by a pass-band Butterworth digital filter, with zero-phase distortion. It is a sixth order filter, with $f_L = 5$ Hz and $f_H = 15$ Hz: this is done because P and T waves have the lowest frequencies of the ECG band (below 5 Hz), and thereby only the frequency content related to the QRS complex is preserved. In addition, the other cut-off frequency (15 Hz) allows to remove prospective sources of noise (like movement artifact or 50 Hz power line noise) which would likely add spikes in the signal, thereby increasing the changes of spurious peak detection;
- **derivative filter.** A derivative filter is a filter approximating differentiation. As such, swift change in slope, even tiny in magnitude, are magnified, while slow trends are flattened. Thus, the final result is that the QRS complex, which has fast slope variations, is now markedly located against an almost flat signal elsewhere. The derivative filter is

implemented as a digital filter having the transfer function

$$H(z) = \frac{f_s}{8} (z^2 + 2z - 2z^{-1} - z^{-2}). \quad (3.10)$$

As appears from (3.9), the filter is non-causal and needs at least two samples forward to implement the derivative. This does not raise any issue since the signal is always given as a finite short segment, which is first temporarily stored and then processed.

- **signal squaring.** Signal point-wise squaring makes all data points positive. In addition non-linear amplification enhances the output of the derivative stage, emphasizing the highest frequencies of the already reduced band;
- **moving window integration.** The signal at this stage is positive definite, with the QRS complexes clearly spotted. Yet, it preserves still high frequency content. To reduce the frequency content leaving unchanged the amplitude level of the signal, the signal at this stage is convoluted with a moving square wave with a width of 100 ms (the maximum QRS duration), and finally normalized in amplitude.

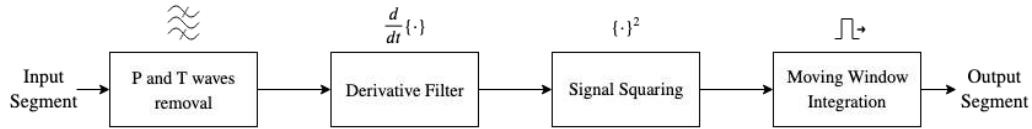


Figure 3.13: Signal preprocessing pipeline.

3.3.4 Adaptive Threshold

The transformed signal is now suited to locate the locations of R peaks. First of all, local maxima are found in the signal using the `find_peaks` function from the SciPy python library, with the constrain that the interval between two consecutive ones lasts 200 ms at least. This duration coincides with the ECG refractory period, i.e. the minimal time interval elapsed from an R peak before another one can occur. This procedure provides the so called *fiducial marks* (Figure 3.14), i.e. candidates for R peak locations [38]. In fact, all R peaks are local maxima of the transformed signal, but the converse is not necessarily true. The search process to discriminate

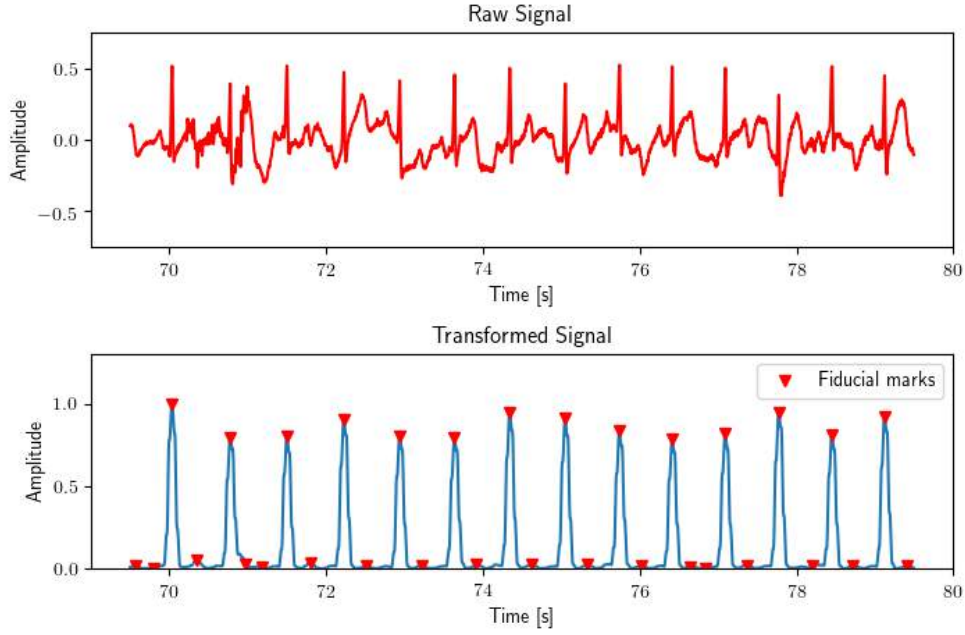


Figure 3.14: Transformed signal of 10 s long segment with fiducial marks.

R peaks is divided in two separate stages: *direct search* stage and *search back* stage. The first consist of a main loop which iterates through all the peak candidates: for every candidate, signal levels are calculated to produce a new threshold value with which the next candidate will be compared against and classified either as an R location, or not. Levels are two: noise level (NL) and signal level (SL). These two values are initialized from the transformed signal as

$$\begin{cases} SL = \text{mean}\{\tilde{s}[n]\} & 0 \leq nT_s < 0.8 \text{ s}, \\ NL = \frac{1}{2} \cdot SL, \end{cases} \quad (3.11a)$$

$$(3.11b)$$

where $T_s = 1/f_s = 1 \text{ ms}$. Then, the threshold is given by

$$TH = C_{TH} \left(\frac{3}{4} \cdot NL + \frac{1}{4} \cdot SL \right), \quad (3.12)$$

where $C_{TH} = 1$ during the direct search stage. Then, the first peak candidate is compared to the threshold: if it exceeds the threshold, then it is classified as an R peak and SL is updated to

a new value. Else, it is classified as non-R peak and NL is updated. The update equations are

$$\begin{cases} NL = C_{\text{noise}} \cdot p[k] + (1 - C_{\text{noise}}) \cdot NL & p[k] \leq TH, \\ SL = C_{\text{noise}} \cdot p[k] + (1 - C_{\text{noise}}) \cdot SL & p[k] > TH, \end{cases} \quad (3.13)$$

$$(3.14)$$

where $C_{\text{noise}} = 0.125$ is a noise coefficient. Being these so as updated, they allow to calculate the next threshold value by (3.12). This process is repeated for all peak candidates. It must be stressed out that, after the first R peak is found, all the successive candidates that exceed the current threshold must undergo a further control. In fact, if the current R peak candidate has occurred within an interval the half of the mean RR interval (calculated over the last 9 elements of $RR[k]$), then this is not classified as an R peak, and NL is updated. Else, the peak is finally recognized as an R peak, and SL is subsequently updated. For the first R peak detections of the current segment, this mean value must be calculated also using former R locations, correspondent to previous segments selected by the sliding window. Despite, when the $RR[k]$ is empty or still lacks of 9 elements (e.g. when the input segment is the first one provided by the sliding window) the said value is set to a standard average RR duration.

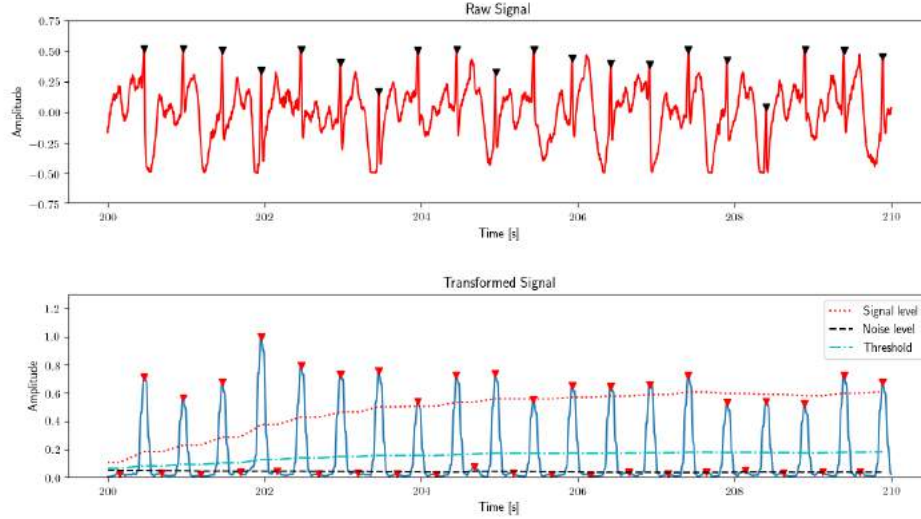


Figure 3.15: Levels and adaptive threshold.

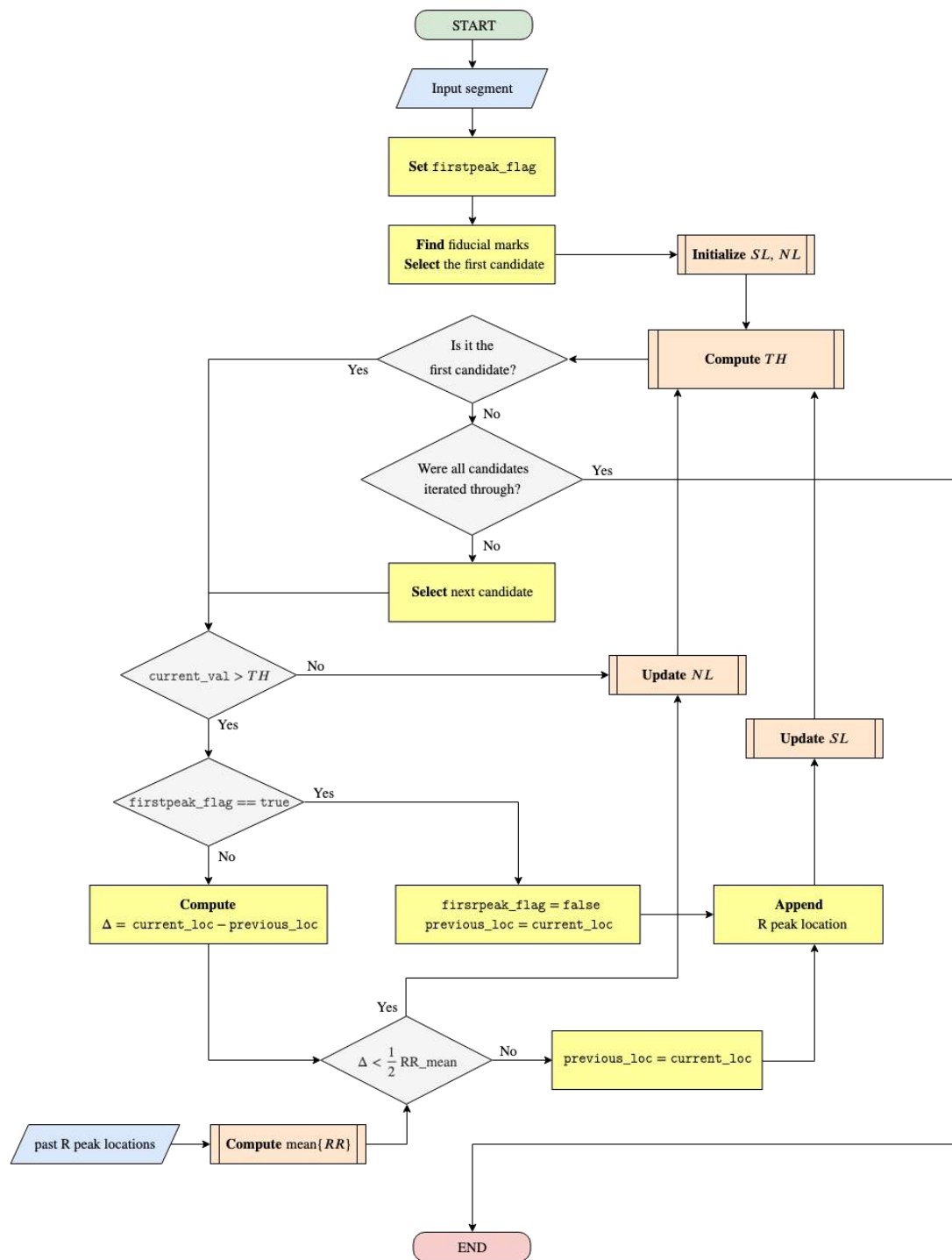


Figure 3.16: Flowchart of the *direct search* stage.

When all candidates have been iterated through, the loop is exited and the direct phase (whose flowchart is depicted in detail in Figure 3.16) is over. Then the algorithm passes to the search back stage (flowchart in the figure below). As opposed to the control devised for too small elapsed time between two consecutive detected peaks in the direct stage, this one is necessary to eventually spot long RR intervals. This is usually due to a failure detection of an R peak, leading to such abnormal RR interval duration. This phase easily consist of redoing the direct stage immediately as one RR interval is found to exceed of a 1.5 factor the mean RR interval, with a threshold coefficient set to $C_{TH} = 0.5$. This choice relies on the fact that the amplitude of one or more undetected R peaks likely lay just below the threshold, which, being now halved by means of C_{TH} , should succeed in finding those missed R peaks.

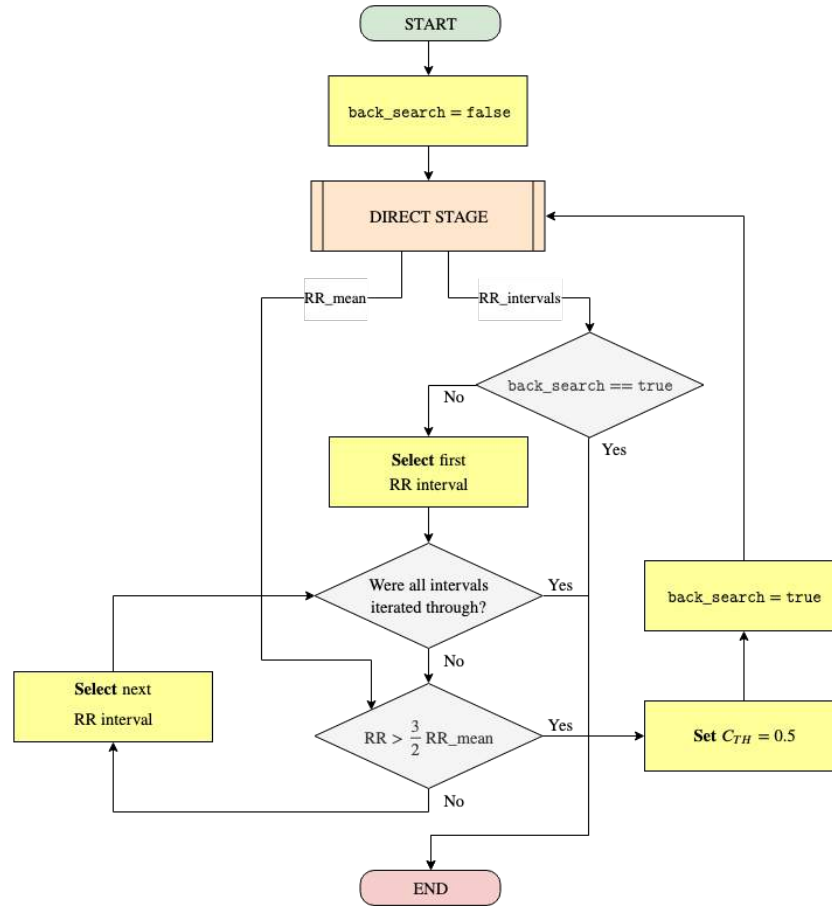


Figure 3.17: Flowchart of the *search back* stage.

The described algorithm finds R peaks on a succession of input segments, differently from the former algorithm proposed by Pan and Tompkins, which works on the other hand on the whole signal without further subdivision. The proposed approach has therefore the merit of adapting the algorithm for real-time BPM computation (through equation (3.9)). Although, the sliding window approach places a limitation on the BPM's refreshing rate in real-time usage. This limitation is peer to 1 s, the sliding step of the window, and therefore both the *Scope* panel window and BPM update rate are bound to (at least) one second when in acquisition mode. In addition to providing real-time BPMs, mainly used to regulate physical exercise during the recordings, the algorithm also manages to save into an array all R peak locations (expressed in seconds) associated to the whole ECG track. In fact, as the window keeps sliding, the algorithm makes sure to save all peaks found for every segment, canceling then repetitions and saving finally all absolute heartbeat locations in the annotation file related to the trial.

Chapter 4

Modeling

Consistently with the established aims of this thesis, in addition to tools and algorithms to obtain synchronized audio/ECG data – and therefore to associate audio segments to their corresponding BPM labels –, it is necessary to introduce computational models capable of producing predictions using such data. Herein, support vector machine algorithms are described in detail, along with the methodologies used to process and structure data into training instances, which are then fed to models. Common techniques for model selection and model evaluation are also discussed, as well as the feature space mapping applied to audio segments through the openSMILE toolkit, one of the most broadly used large space feature extractor in speech and music. An overview of the stages pursued in order to build computational models (comprehensive of data collection) is presented in Figure 4.1.

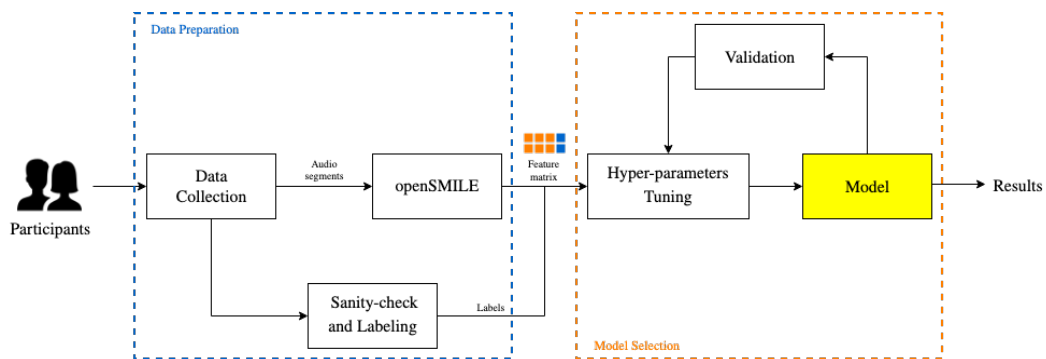


Figure 4.1: Main stages of the study.

4.1 Support Vector Machine

Support Vector Machine (SVM) is one of the most popular machine learning algorithm for supervised learning. It conceptually implements the following idea: input vectors are (non-linearly) mapped to a very high-dimension feature space. In this feature space a linear decision surface is constructed in such a way that the margin of separation between positive and negative examples is maximized [57, 58]. SVM has established itself as the most widely used kernel-learning algorithm. Indeed, SVMs represent the state-of-the-art by virtue of their good generalization performance, relative ease of use, and rigorous theoretical foundations. Moreover, in a practical context, they are capable of delivering robust performance in solving pattern-recognition and regression problems, especially in learning problems with small datasets. However, the major limitation of SVMs is the fast increase in their computing and storage requirements with respect to the number of training examples. These severe requirements tend to leave many large-scale learning problems beyond the reach of SVMs [23]. Consider a set of samples $\mathfrak{D} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$, where $\mathbf{x}_i \in X$ is the input vector and d_i is the corresponding label (target output), with $y_i = 1$ representing positive class samples and $y_i = -1$ representing negative class samples. In this case with only two classes, input vectors can be separated by a decision surface, usually expressed as the locus of points satisfying an equation of the form $f(\mathbf{x}) = 0$. If f is linear, then the decision surface is a hyperplane of the input space X .

Proposition. *A dataset $\mathfrak{D} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$ is said **linearly separable** if, given a decision surface in the form*

$$\mathbf{w} \cdot \mathbf{x} + c = 0, \quad (4.1)$$

it holds that

$$\begin{cases} \mathbf{w} \cdot \mathbf{x}_i + c > 0 & \text{if } y_i = 1, \\ \mathbf{w} \cdot \mathbf{x}_i + c < 0 & \text{if } y_i = -1. \end{cases} \quad (4.2a)$$

$$(4.2b)$$

*The **decision rule** will be thus*

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + c, \quad (4.3)$$

on which is taken the sign to implement simultaneously equations (4.2a), (4.2b).

Proof. The parametric equation of a hyperplane H in a n -dimensional space is given by a vector \mathbf{x}_0 and n spanning vectors $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$, namely

$$\mathbf{x} = \mathbf{x}_0 + \alpha_1 \mathbf{v}_1 + \dots + \alpha_n \mathbf{v}_n. \quad (4.4)$$

Given \mathbf{w} orthogonal to each of the spanning vectors $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$, using scalar multiplication between (4.4) and \mathbf{w} gives

$$\mathbf{w} \cdot \mathbf{x} = \mathbf{w} \cdot \mathbf{x}_0 + \underbrace{\alpha_1 \mathbf{w} \cdot \mathbf{v}_1 + \dots + \alpha_n \mathbf{w} \cdot \mathbf{v}_n}_{=0}. \quad (4.5)$$

Putting $\mathbf{w} \cdot \mathbf{x}_0 = -c$, one obtains (4.1). This equation describes the distance of a point $\mathbf{x} \in X$ from the hyperplane: trivially, any point with zero distance will belong to it. Choosing this as a separation surface, if $\mathbf{w} \cdot \mathbf{x}_i + c$ is positive, then \mathbf{x}_i belongs to the upper semi-space individualized by the decision surface, therefore $y_i = 1$, yielding (4.2a); vice versa, if $\mathbf{w} \cdot \mathbf{x}_i + c$ is negative, then \mathbf{x}_i belongs to the lower semi-space individualized by the decision surface, therefore $y_i = -1$, yielding (4.2b). \square

Equations (4.2a), (4.2b) are sufficient to implement a decision rule for a linearly separable set. Nonetheless, the issue is that the separation hyperplane described by (4.1) does not imply any principle of optimality. Consider for example $X \equiv \mathbb{R}^2$ in Figure 4.2: all three separation lines implement correctly a classification rule, but only one maximizes the margin of separation between the nearest positive and negative examples: this is the idea at the basis of SVM. The first step for the derivation of \mathbf{w} and c following the maximum separation margin, is to enforce the decision rules (4.2a), (4.2b), by imposing further that any example must be classified beyond a certain value δ from the decision boundary, that is

$$\begin{cases} \mathbf{w} \cdot \mathbf{x}_i + c \geq \delta & \text{if } y_i = 1, \\ \mathbf{w} \cdot \mathbf{x}_i + c \leq -\delta & \text{if } y_i = -1. \end{cases} \quad (4.6a)$$

$$(4.6b)$$

It should be noted that the vector \mathbf{w} is unique for the dataset \mathcal{D} by definition. In fact, it is the only vector orthogonal at the same time to every vector in the set $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$. Conversely, \mathbf{x}_0 defines the origin where the said set of vectors is placed, and it is easy to see that every value

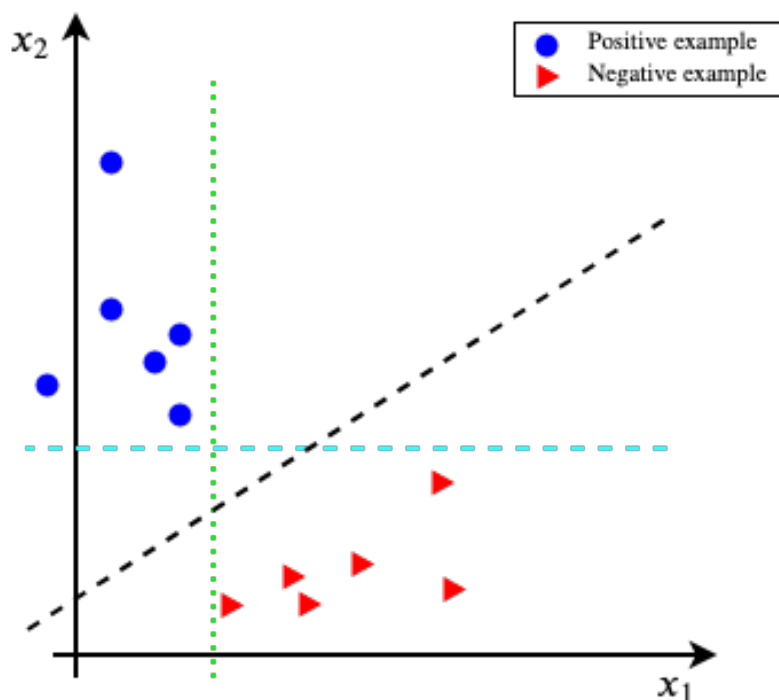


Figure 4.2: Three different decision boundaries for a linearly separable dataset. The black one satisfies the maximum margin principle.

$\mathbf{x}_0 \in H$ is admissible to parameterize the hyperplane by (4.4). This means that $c = -\mathbf{w} \cdot \mathbf{x}_0$ is arbitrary too. To avoid this arbitrariness, c is set to $b + \delta - 1$ for (4.6a) and to $b - \delta + 1$ for (4.6b). With this choice, the constraint equations become

$$\begin{cases} \mathbf{w} \cdot \mathbf{x}_i + b \geq 1 & \text{if } y_i = 1, \\ \mathbf{w} \cdot \mathbf{x}_i + b \leq -1 & \text{if } y_i = -1. \end{cases} \quad (4.7a)$$

$$\quad \quad \quad \begin{cases} \mathbf{w} \cdot \mathbf{x}_i + b \leq -1 & \text{if } y_i = -1. \end{cases} \quad (4.7b)$$

For mathematical convenience, equations (4.7a), (4.7b) can be lumped together in a single inequality. In fact, it can be written that

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \quad i = 1, \dots, m. \quad (4.8)$$

It is easy to show by simple substitution of $y_i = \pm 1$ that (4.8) satisfies both (4.7a) and (4.7b). Now, the quantity to minimize must be computed: to this end, consider the pair of examples

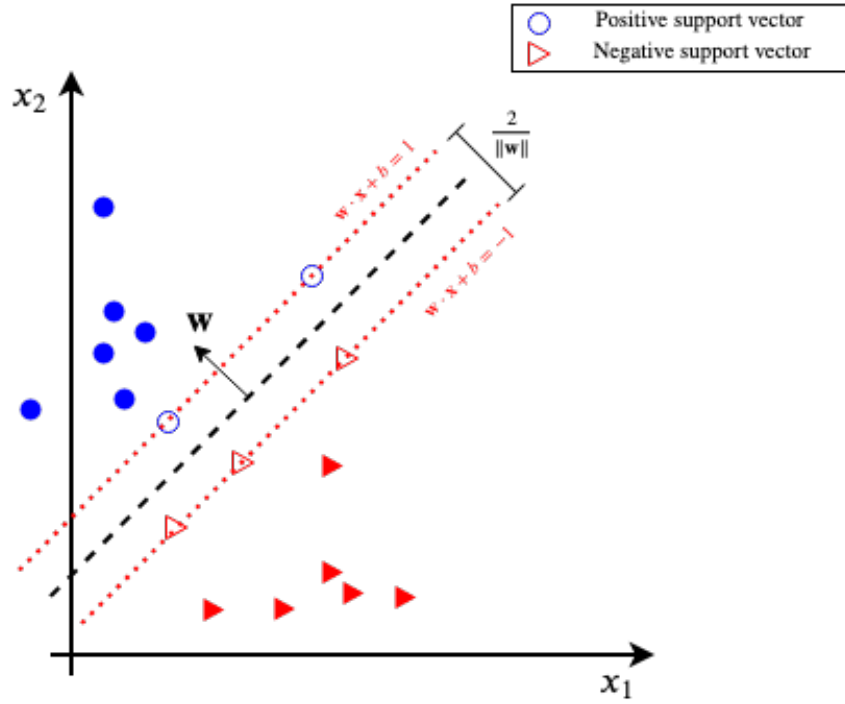


Figure 4.3: Maximum margin separation decision boundary and support vectors.

$\mathbf{x}_+^{(s)}$ and $\mathbf{x}_-^{(s)}$ satisfying respectively (4.7a) and (4.7b) with the equality sign, namely

$$\begin{cases} \mathbf{w} \cdot \mathbf{x}_+^{(s)} + b = 1, & (4.9a) \\ \mathbf{w} \cdot \mathbf{x}_-^{(s)} + b = -1. & (4.9b) \end{cases}$$

These vectors, $\mathbf{x}_+^{(s)}$ and $\mathbf{x}_-^{(s)}$, and in general any vector of the input space X satisfying one between (4.7a) and (4.7b) with the equality sign, are called **support vectors**, as shown in Figure 4.3. Then, the margin ρ is the difference between these two projected along the direction of \mathbf{w} , and it reads

$$\rho = (\mathbf{x}_+^{(s)} - \mathbf{x}_-^{(s)}) \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|} = \frac{\mathbf{w} \cdot \mathbf{x}_+^{(s)} - \mathbf{w} \cdot \mathbf{x}_-^{(s)}}{\|\mathbf{w}\|} = \frac{1 - b - (-1 - b)}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}. \quad (4.10)$$

To be noted, maximizing the expression for the margin ρ is the same of minimizing the norm $\|\mathbf{w}\|$. Therefore, the objective function chosen is half the squared norm of $\|\mathbf{w}\|$ [8], in order to

cleverly formulate the optimization problem under the umbrella of convex optimization, avoiding thus to get stuck with local maxima. The optimization problem now reads as

$$\begin{cases} \min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|^2, \\ \text{sub } y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0, \end{cases} \quad (4.11)$$

for $i = 1, \dots, m$. This is a constrained optimization problem, with as many constraints as the number of input vectors \mathbf{x}_i . The Lagrangian can be written as

$$\mathcal{L}(\mathbf{w}, b, \lambda_1, \dots, \lambda_m) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^m \lambda_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1], \quad (4.13)$$

where $\lambda_1, \dots, \lambda_m$ are the Lagrangian multipliers. This is the **primal problem**, whose solution lies in a saddle point in a $n + m + 1$ dimension space. In order to find it, one needs to explicit partial derivatives of \mathcal{L} with respect to \mathbf{w} and b and make them equal to zero:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^m \lambda_i y_i \mathbf{x}_i = 0, \quad (4.14)$$

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_{i=1}^m \lambda_i y_i = 0. \quad (4.15)$$

From condition (4.14), it is possible to express \mathbf{w} as

$$\mathbf{w} = \sum_{i=1}^m \lambda_i y_i \mathbf{x}_i. \quad (4.16)$$

Expanding (4.13) and using (4.14) and (4.15), one can write

$$\begin{aligned} \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^m \lambda_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] &= \frac{1}{2} \mathbf{w} \cdot \mathbf{w} - \sum_{i=1}^m \lambda_i y_i \mathbf{w} \cdot \mathbf{x}_i - b \overbrace{\sum_{i=1}^m \lambda_i y_i}^{=0} + \sum_{i=1}^m \lambda_i = \\ &= \frac{1}{2} \sum_{i=1}^m \lambda_i y_i \mathbf{x}_i \cdot \sum_{j=1}^m \lambda_j y_j \mathbf{x}_j - \sum_{i=1}^m \lambda_i y_i \left(\sum_{j=1}^m \lambda_j y_j \mathbf{x}_j \right) \cdot \mathbf{x}_i + \sum_{i=1}^m \lambda_i = \\ &= \sum_{i=1}^m \lambda_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \lambda_i \lambda_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j = Q(\lambda). \end{aligned} \quad (4.17)$$

The latter quantity $Q(\boldsymbol{\lambda})$, called the dual Lagrange function, is the new objective function depending on the vector of multipliers $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_m)$, and it is used to define the so called **dual problem**, which is based on the constrained optimization of Q rather than \mathcal{L} , as long as the conditions provided **duality theorem** are ensured [5], as enounced in the following statement.

Proposition. *Given the dataset $\mathfrak{D} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$, find the Lagrange multipliers $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_m)$ that maximize the objective function*

$$Q(\boldsymbol{\lambda}) = \sum_{i=1}^m \lambda_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \lambda_i \lambda_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j, \quad (4.18)$$

subject to the constraints

$$\begin{cases} \sum_{i=1}^m \lambda_i y_i = 0, \\ \lambda_i \geq 0 \end{cases} \quad \text{for } i=1, \dots, m. \quad (4.19)$$

$$(4.20)$$

The **Kuhn-Tucker theorem** plays an important part in the theory of constrained optimization [18, 29]. According to this theorem, at the saddle point of \mathcal{L} in $(\mathbf{w}_0, b_0, \boldsymbol{\lambda}_0)$, any Lagrange multiplier $\lambda_{0,i}$ and its corresponding constraint are connected by an equality like

$$\lambda_{0,i} [y_i (\mathbf{x}_i \cdot \mathbf{w}_0 + b_0)] = 0. \quad (4.21)$$

Since $\boldsymbol{\lambda}$ cannot be the null vector (at most, some of its elements equals zero for some indexes, being the other greater than 0), (4.21) is satisfied by $y_i (\mathbf{x}_i \cdot \mathbf{w}_0 + b_0)$ being zero. By definition, support vectors are those satisfying this latter condition, thus non-zero multipliers are only those associated with support vectors, being all the other multipliers null. Calling $\lambda_i^{(s)}$ the non-zero Lagrange multipliers allows to rewrite (4.16) as

$$\mathbf{w}_0 = \sum_{i=1}^{m_s} y_i \lambda_i^{(s)} \mathbf{x}_i^{(s)}. \quad (4.22)$$

That allows to affirm that the optimal \mathbf{w}_0 is given by a linear combination of support vectors. This is an improvement of (4.16), since it is possible to calculate \mathbf{w}_0 simply from a limited number of training examples (the support vectors is a subset of \mathfrak{D}) and the associated multipliers. Then,

it is easy to calculate b_0 by choosing a positive support vector and combining (4.22) with (4.9a):

$$b_0 = 1 - \mathbf{w}_0 \cdot \mathbf{x}_+^{(s)} = 1 - \left(\sum_{i=1}^{m_s} y_i \lambda_i^{(s)} \mathbf{x}_i^{(s)} \right) \cdot \mathbf{x}_+^{(s)} = 1 - \sum_{i=1}^{m_s} y_i \lambda_i^{(s)} \mathbf{x}_i^{(s)} \cdot \mathbf{x}_+^{(s)}. \quad (4.23)$$

The solution (\mathbf{w}_0, b_0) is *sparse*, meaning that only a small number of support vectors are needed to actually compute the decision surface. Despite this, the support vectors are not *a priori* known, and can be identified looking for non-zero multipliers only after the dual problem is solved.

4.1.1 Soft-margin Decision Surface

The discussion so far has focused on linearly separable patterns. It is more likely that given a dataset \mathfrak{D} , it is not possible to construct a separating hyperplane without encountering classification errors or causing overfitting. It may happen that a dataset contains a couple of outliers from clearly linearly separable patterns. Nonetheless, those outliers would make technically the dataset non-linearly separable, and the employment of highly non-linear decision boundary would not generalize well with unseen data. Therefore, it is useful to introduce a regularization term to this end. Observe that a margin of separation between classes is said to be soft if a given sample (\mathbf{x}_i, y_i) violates the condition expressed in (4.8). This violation can arise in one of the ways:

1. the example (\mathbf{x}_i, y_i) falls inside the region of separation, but on the correct side of the decision surface (Figure 4.4a);
2. the example (\mathbf{x}_i, y_i) falls on the wrong side of the decision surface (Figure 4.4b).

To set the stage for a formal treatment of non-separable data points, a new set of non negative variables, called **slack variables**, are introduced. Then, the constraints equation (4.8) is modified accordingly as

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i. \quad (4.24)$$

In some extreme case of linearly separable patterns, where the margin of separation is very tiny, it is advisable to use this approach and allow a certain number of outliers in the training set instead of forcing the machine to find the separation hyperplane and likely produce way more

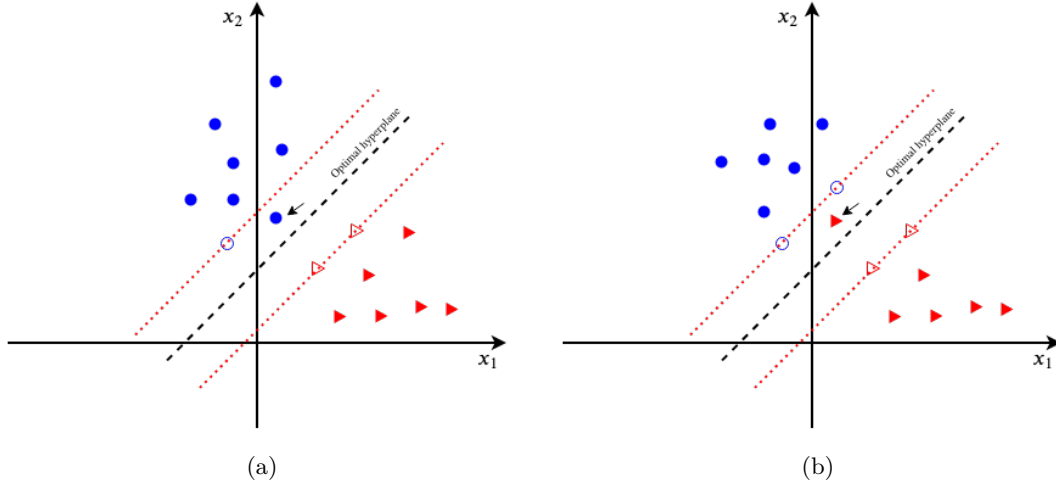


Figure 4.4: An example falls within the separation margin but on the right side of the decision boundary (a). An example falls within the separation margin and on the wrong side of the decision boundary (b).

errors for unseen data (bad generalization). Slack variables measure the deviation of outliers falling in one of the two situation listed above: in fact, for $0 < \xi_i \leq 1$, the data point falls inside the region of separation, but on the correct side of the decision surface; for $\xi_i > 1$, it falls on the wrong side of the separating hyperplane. For all the other points, $\xi_i = 0$. To weight the influence of slack variables $\boldsymbol{\xi} = (\xi_1, \dots, \xi_m)$ on the optimization problem, the augmented optimization function can be written as

$$J(\mathbf{w}, \boldsymbol{\xi}) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m \xi_i, \quad (4.25)$$

subjected to constraints

$$\begin{cases} y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \\ \xi_i \geq 0. \end{cases} \quad (4.26)$$

$$(4.27)$$

This is also called L1-SVM, since the slack variables are summed without squaring. The parameter C controls the trade-off between complexity of the machine and the number of non-separable point, and for this reason is called the **complexity parameter**. When the parameter C is assigned a large value, the implication is high confidence in the quality of the training set \mathfrak{T} being constituted by separable patterns. Conversely, when C is assigned a small value, the training set \mathfrak{T} is considered to be noisy or with biased labels with examples likely non clustered. The

Lagrangian would be in this case

$$\begin{aligned} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \lambda_1, \dots, \lambda_m, \gamma_1, \dots, \gamma_m) = & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m \xi_i + \\ & - \left(\sum_{i=1}^m \lambda_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i] + \sum_{i=1}^m \gamma_i \xi_i \right), \end{aligned} \quad (4.28)$$

where $\gamma_1, \dots, \gamma_m$ are the Lagrangian multipliers associated to slack variables. Partial derivatives of the above quantity about \mathbf{w} and b will yield again (4.14) and (4.15), together with

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = C - \lambda_i - \gamma_i = 0 \quad i = 1, \dots, m. \quad (4.29)$$

From the latter condition, if $\lambda_i > C$, then the corresponding multiplier γ_i would be greater than zero, which in general would disagree the Kuhn-Tucker condition (being $\xi_i \geq 0$ by definition). Therefore, it must be imposed that

$$\lambda_i \leq C. \quad (4.30)$$

Then, using conditions (4.14), (4.15) and (4.29) allow to simplify and obtain Q exactly in the same form of (4.18). Indeed, the dual problem can be stated as well for soft-margin SVM.

Proposition. *Given the dataset $\mathcal{D} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$, find the Lagrange multipliers $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_m)$ that maximize the objective function*

$$Q(\boldsymbol{\lambda}) = \sum_{i=1}^m \lambda_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \lambda_i \lambda_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j, \quad (4.31)$$

subject to the constraints

$$\begin{cases} \sum_{i=1}^m \lambda_i d_i = 0, \\ 0 \leq \lambda_i \leq C \end{cases} \quad \text{for } i = 1, \dots, m. \quad (4.32)$$

$$(4.33)$$

Concretely, the $\boldsymbol{\lambda}_0$ is found through iterative methods like stochastic gradient descent (SDG) [50]. To be noted, the dual problem for the soft-margin SVM is identical to the one for *hard* SVM except for the range of Lagrange multipliers, which in this case cannot exceed C .

4.1.2 The Kernel Method

In both linearly separable hyperplane and soft-margin methods, the margin of separation problem consisted in finding the Lagrange multipliers (under the constraints) maximizing the objective function of the dual problem. To be noted, the dual problem is a constrained optimization problem with m variables to determine (Lagrangian multipliers) instead of $n + 1$, coming from the couple (\mathbf{w}, b) . In many practical cases, the input space dimension is great such that $n + 1 \gg m$, and therefore the sparse solution found through Lagrange multipliers is one of the reason why SVM is a powerful tool in case of high feature space and small amount of data. Nonetheless, a separation hyperplane, even with the slack variables approach, may result in poor generalization if the patterns of the training set are not linearly separated as shown in Figure 4.5.

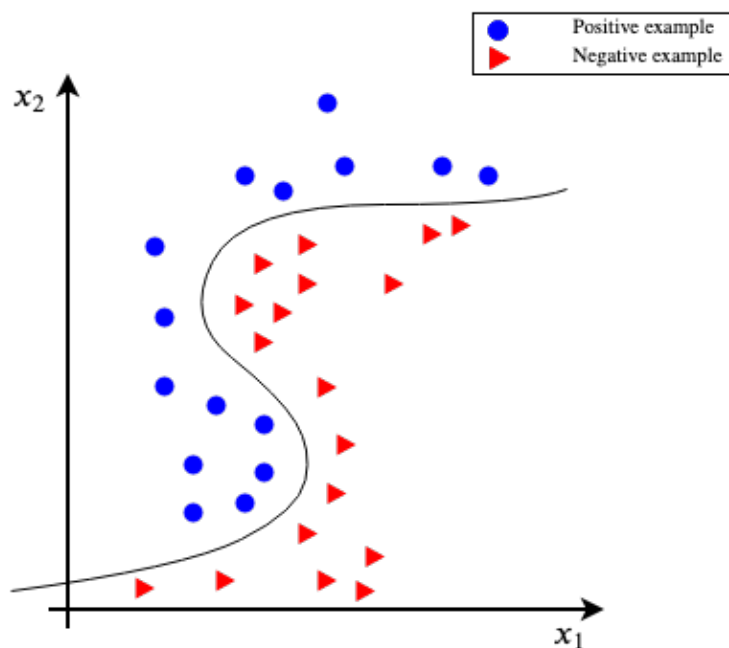


Figure 4.5: Non-linearly separable patterns.

It is noteworthy that quantities in equations (4.18) and (4.31) depend only on dot products of input vectors: this observation is at the basis of the so called **kernel method**. Consider a dataset \mathcal{D} with non-linearly separable patterns. To apply the formalism of maximum margin

separation hyperplane, one has to transform the n -dimensional input space X into a higher N -dimensional feature space F with a vectorial function $\phi : X \rightarrow F$, where the input vectors become linearly separable. Then, a N -dimensional (even infinite) hyperplane is constructed for the set of transformed vectors

$$\phi(\mathbf{x}_i) = (\phi_1(\mathbf{x}_i), \dots, \phi_N(\mathbf{x}_i)) \quad i = 1, \dots, m. \quad (4.34)$$

In this way, classification of an unseen vector \mathbf{x} is done by first transforming it with ϕ and then taking the sign of the decision rule, that now reads

$$f(\mathbf{x}) = \phi(\mathbf{x}) \cdot \mathbf{w}_0 + b_0. \quad (4.35)$$

Recalling that the weight vector \mathbf{w}_0 is given by a linear combination of support vectors, one can extend the idea to support vectors expressed in feature space, having

$$\mathbf{w}_0 = \sum_{i=1}^{m_s} y_i \lambda_i^{(s)} \phi(\mathbf{x}_i^{(s)}). \quad (4.36)$$

Similarly for b , it reads

$$b_0 = 1 - \sum_{i=1}^{m_s} y_i \lambda_i^{(s)} \phi(\mathbf{x}_i^{(s)}) \cdot \phi(\mathbf{x}_+^{(s)}). \quad (4.37)$$

Feeding equations (4.36), (4.37) in (4.35), the decision rule can be rewritten only in function of dot products of transformed input vectors as

$$f(\mathbf{x}) = \sum_{i=1}^{m_s} y_i \lambda_i^{(s)} \boxed{\phi(\mathbf{x}_i^{(s)}) \cdot \phi(\mathbf{x})} - \sum_{i=1}^{m_s} y_i \lambda_i^{(s)} \boxed{\phi(\mathbf{x}_i^{(s)}) \cdot \phi(\mathbf{x}_+^{(s)})} + 1. \quad (4.38)$$

Following a similar reasoning, the objective function of the dual problem is expressed as

$$Q(\boldsymbol{\lambda}) = \sum_{i=1}^m \lambda_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \lambda_i \lambda_j y_i y_j \boxed{\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)}. \quad (4.39)$$

Equations (4.39), (4.38) are respectively needed for training (finding the multipliers and their related support vectors) and prediction (taking the sign of $f(\mathbf{x})$). As already mentioned, they can be computed directly knowing the dot product of a couple transformed vectors. This operation

defines a non-linear function defined on any pair of vectors (\mathbf{u}, \mathbf{v}) of X , which is called **kernel**.

Definition. A kernel is a function $K(\cdot, \cdot)$ that computes the inner product of the images produced in feature space F , under the embedding ϕ , of two input vectors \mathbf{u}, \mathbf{v} :

$$K(\mathbf{u}, \mathbf{v}) := \phi(\mathbf{u}) \cdot \phi(\mathbf{v}). \quad (4.40)$$

It is evident that the knowledge of the kernel K instead of the N components of ϕ allows to effectively implement kernel-SVMs. In fact, insofar as pattern classification in the output space is concerned, specifying the kernel $K(\cdot, \cdot)$ is sufficient, without the need of explicitly compute the weight vector \mathbf{w}_0 . It is possible to represent kernel-SVM with a network architecture, such as the one depicted in Figure 4.6 as a single-layer network called support vector network (as originally conceived by Vapnik et al. [57, 58]).

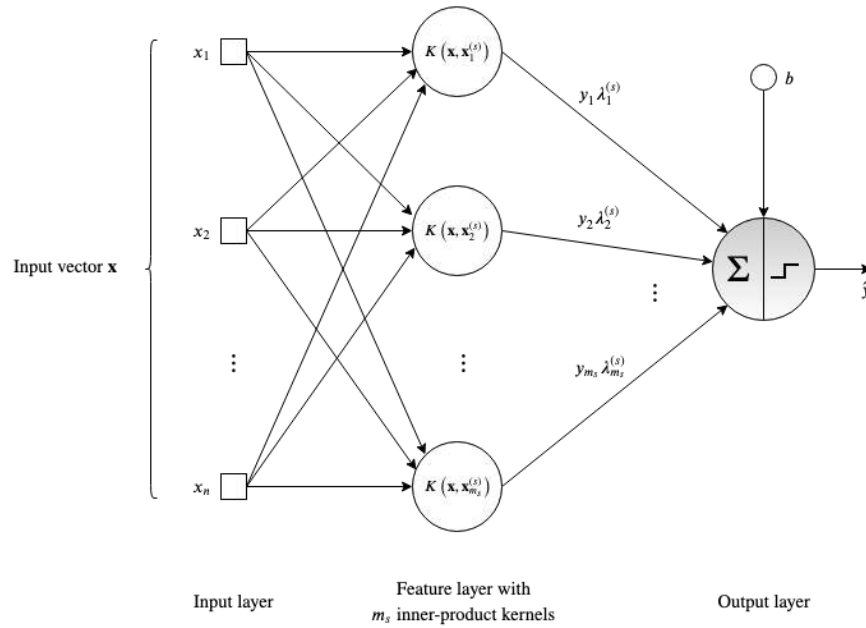


Figure 4.6: Support vector network.

Not any function K is eligible for a kernel. In first place, symmetry of K must be ensured, that is $K(\mathbf{u}, \mathbf{v}) \equiv K(\mathbf{v}, \mathbf{u})$. Then, a kernel candidate must abide by the Mercer's theorem [34], under which a kernel function K is actually decomposable in terms of a set (eventually infinite) of ϕ_j ,

which are the eigenfunctions of K . Within these requirements, it is possible to choose a feasible kernel to implement an SVM. In Table 4.1, three common types of kernels are summarized: polynomial kernel, radial basis function kernel and hyperbolic tangent kernel. For the first two, their respective parameters p and σ must be *a priori* specified. For the sigmoid kernel, there is actually a finite set of β_0, β_1 values which make this kernel compliant with the Mercer's theorem [23].

Type of kernel	Expression of $K(\cdot, \cdot)$
Polynomial kernel	$K(\mathbf{u}, \mathbf{v}) = (1 + \mathbf{u} \cdot \mathbf{v})^p$
RBF kernel	$K(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{\ \mathbf{u} - \mathbf{v}\ ^2}{2\sigma^2}\right)$
Sigmoid kernel	$K(\mathbf{u}, \mathbf{v}) = \tanh(\beta_0 \mathbf{u} \cdot \mathbf{v} + \beta_1)$

Table 4.1: Three among the most used kernels in SVM problems.

Regardless of how a SVM is implemented, it differs from the conventional approach to the design of multilayer perceptron in a fundamental way. In the conventional approach, model complexity is controlled by keeping the number of features (i.e. hidden layers) small. On the other hand, SMVs offer a solution to the design of a learning machine by controlling the model complexity independently of dimensionality [47, 57].

4.1.3 Support Vector Regressor

So far SVM was presented as a binary classification machine learning method. However, SVM can be adapted also to regression and time-series prediction problems. To set the stage for describing the so called ε -SV, linear regression case will be considered first, and then extended to non-linear regression. The principle is the following: find a fitting function $f : X \rightarrow Y$ (where the output space Y of targets d_i usually coincides with a subset of \mathbb{R}) that has at most deviation peer to ε from the targets d_i for all the training data, and at the same time is as flat as possible. In other words, \mathbf{w}_0 and b_0 are determined such that any error $e_i = d_i - y_i$ (with $y_i = f(\mathbf{x}_i)$) is smaller than ε , which is a precision parameter and must be *a priori* specified

[23, 55]. To expose calculations with geometrical insight, consider $X = Y \equiv \mathbb{R}$ and a dataset $\mathfrak{D} = \{(x_1, d_1), \dots, (x_m, d_m)\}$. The goal is to find a regression line following the margin intuition already used for classification. This latter on will be expressed by the formula of line in 2D space written in explicit form

$$y = \alpha + \beta x. \quad (4.41)$$

Then, $X \times Y$ is a vectorial space and it is possible to consider vectors whose components are given by elements of X and Y . In the case at hand, this will be $\tilde{\mathbf{x}} = (x, y)$. By introducing this, (4.41) can be expressed in same form of (4.1). In fact, posing $\tilde{\mathbf{w}} = (\beta, -1)$ and $b = \alpha$, one can write

$$\beta x - y + \alpha = 0 \implies \tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}} + b = 0. \quad (4.42)$$

It should be born in mind that b in this case is the intercept with the vertical axis. Moreover, it is easy to see that $\tilde{\mathbf{w}}$ in the obtained form is orthogonal to the line but pointing on the negative semi-plane of $X \times Y$. Similarly as in §4.3, the two gutter lines, parallel to the regression line, are defined. In particular, the requirement is that any regression error in its absolute value results smaller than ε , hence the confining gutters are easily obtained adding or subtracting ε to the intercept, that is:

$$\begin{cases} \tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}} + b + \varepsilon = 0, & (4.43a) \\ \tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}} + b - \varepsilon = 0. & (4.43b) \end{cases}$$

Then, calculating the margin ρ between the gutters follows the geometric intuition depicted in Figure 4.6. In fact, given two points $\tilde{\mathbf{x}}_{\text{upper}}$ and $\tilde{\mathbf{x}}_{\text{lower}}$ belonging respectively to these lines expressed by (4.43a), (4.43b), the projection of their difference on the direction of $\tilde{\mathbf{w}}$ yields

$$\rho = (\tilde{\mathbf{x}}_{\text{lower}} - \tilde{\mathbf{x}}_{\text{upper}}) \cdot \frac{\tilde{\mathbf{w}}}{\|\tilde{\mathbf{w}}\|} = \frac{\tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}}_{\text{lower}} - \tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}}_{\text{upper}}}{\|\tilde{\mathbf{w}}\|} = \frac{\varepsilon - b - (-\varepsilon - b)}{\|\tilde{\mathbf{w}}\|} = \frac{2\varepsilon}{\|\tilde{\mathbf{w}}\|}. \quad (4.44)$$

The quantity expressed in (4.44) must be maximized. Observe that the width 2ε is a limiting factor for ρ : in fact, $\|\tilde{\mathbf{w}}\| = \sqrt{\beta^2 + 1}$ equals 1 only if $\beta = 0$ (the regression line is horizontal), and for any other value of m , the margin ρ will be smaller than 2ε . Therefore, the objective is to find the smallest $\tilde{\mathbf{w}}$ such that the margin approaches 2ε under the constraints that the

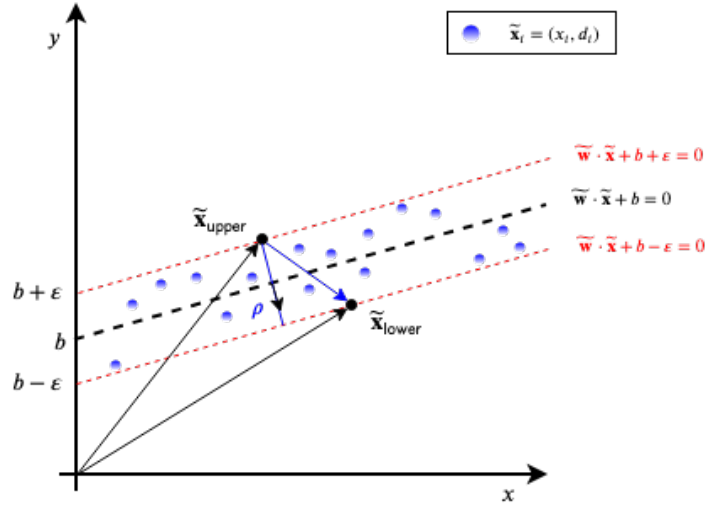


Figure 4.7: Regression line with the maximum margin approach.

admissible errors are such that $|e_i| < \varepsilon$. Thus, finding the regression line can be enounced as convex optimization problem:

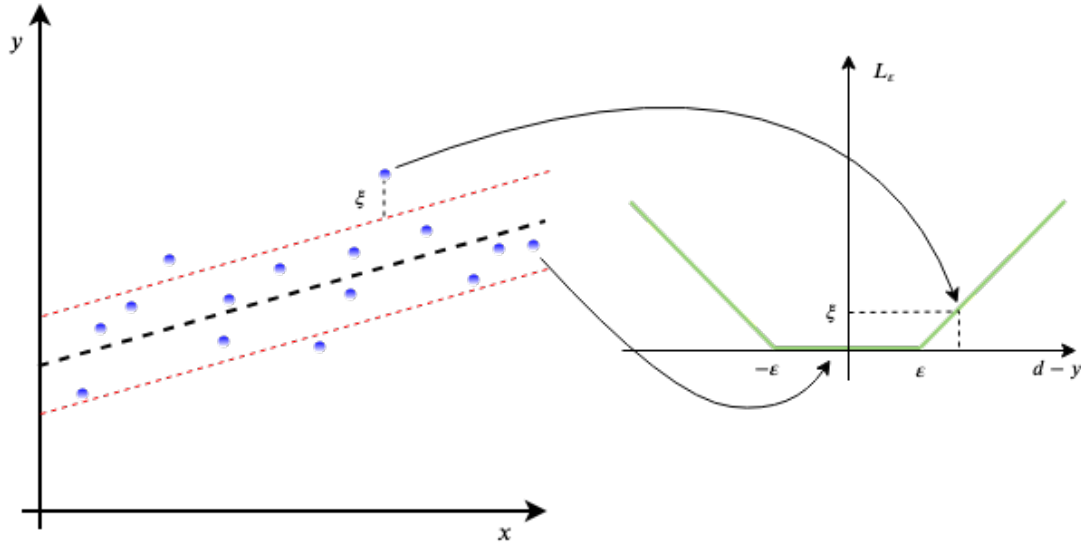
$$\begin{cases} \min_{\tilde{\mathbf{w}}} \frac{1}{2} \|\tilde{\mathbf{w}}\|^2, \\ \text{sub } |d_i - y_i| \leq \varepsilon. \end{cases} \quad (4.45)$$

$$(4.46)$$

The tacit assumption is that it actually exists a regression line that approximates all pairs (x_i, y_i) with ε precision, or in other words, that the convex optimization is feasible. Sometimes, however, this may not be the case and some errors might be allowed. Analogously to the soft margin introduced in §4.1.2, one can introduce also here slack variables ξ_i to cope with otherwise infeasible optimization problems. To quantify slack variables, it is useful to introduce the so called *hinge loss* function

$$L_\varepsilon(d, y) = \xi := \begin{cases} |d - y| - \varepsilon & \text{for } |d - y| \geq \varepsilon, \\ 0 & \text{otherwise.} \end{cases} \quad (4.47)$$

This function is employed to penalize all errors greater (in absolute value) than ε . To continue, it is worthy to consider the problem in the general case for $X \equiv \mathbb{R}^n$. In this case, the $\tilde{\mathbf{w}}$ vector would be made by a block of parameters \mathbf{w} to fit and last entry peer to -1 . Since $\|\tilde{\mathbf{w}}\|^2 = \|\mathbf{w}\|^2 + 1$,

Figure 4.8: ε -insensitive tube and hinge loss function.

the new cost function to optimize can be written using \mathbf{w} without loss of generality:

$$J(\mathbf{w}, \boldsymbol{\xi}) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m (\xi_i + \xi_i^*), \quad (4.48)$$

under constraints

$$\begin{cases} |d_i - \mathbf{w} \cdot \mathbf{x}_i - b| \leq \varepsilon + \xi_i, \\ \xi_i \geq 0. \end{cases} \quad (4.49)$$

$$(4.50)$$

It must be observed that the constrain of (4.49) contains an absolute value, which should be resolved. Being the couple (\mathbf{w}, b) unknown a priori, the sign of $d_i - \mathbf{w} \cdot \mathbf{x}_i - b$ is a priori unknown as well, and the condition with the absolute value carries indeed two conditions. Therefore, a new group of slack variables, ξ_i^* , is introduced, yielding four constraints:

$$\begin{cases} d_i - \mathbf{w} \cdot \mathbf{x}_i - b \leq \varepsilon + \xi_i, \end{cases} \quad (4.51)$$

$$\begin{cases} \mathbf{w} \cdot \mathbf{x}_i + b - d_i \leq \varepsilon + \xi_i^*, \end{cases} \quad (4.52)$$

$$\begin{cases} \xi_i \geq 0, \end{cases} \quad (4.53)$$

$$\begin{cases} \xi_i^* \geq 0. \end{cases} \quad (4.54)$$

The Lagrangian is

$$\begin{aligned} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \lambda_1, \lambda_1^*, \dots, \gamma_1, \gamma_1^* \dots) = & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m (\xi_i + \xi_i^*) + \\ & - \left(\sum_{i=1}^m \lambda_i [\varepsilon + \xi_i + \mathbf{w} \cdot \mathbf{x}_i + b - d_i] + \sum_{i=1}^m \gamma_i \xi_i \right) + \\ & - \left(\sum_{i=1}^m \lambda_i^* [\varepsilon + \xi_i^* - \mathbf{w} \cdot \mathbf{x}_i - b + d_i] + \sum_{i=1}^m \gamma_i^* \xi_i^* \right). \end{aligned} \quad (4.55)$$

Taking derivatives and making them equal to zero will yield the following relationships:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^m (\lambda_i - \lambda_i^*) \mathbf{x}_i = 0 \implies \mathbf{w} = \sum_{i=1}^m (\lambda_i - \lambda_i^*) \mathbf{x}_i, \quad (4.56)$$

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_{i=1}^m (\lambda_i - \lambda_i^*) = 0, \quad (4.57)$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = C - \lambda_i - \gamma_i = 0 \implies \gamma_i = C - \lambda_i, \quad (4.58)$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i^*} = C - \lambda_i^* - \gamma_i^* = 0 \implies \gamma_i^* = C - \lambda_i^*. \quad (4.59)$$

Using equations (4.56), (4.57), (4.58) and (4.59) into (4.55) yield to the usual dual optimization problem, enounced as follow.

Proposition. *Given the dataset $\mathcal{D} = \{(\mathbf{x}_1, d_1), \dots, (\mathbf{x}_m, d_m)\}$, find the Lagrange multipliers $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_m)$, $\boldsymbol{\lambda}^* = (\lambda_1^*, \dots, \lambda_m^*)$ that maximize the objective function*

$$Q(\boldsymbol{\lambda}, \boldsymbol{\lambda}^*) = \sum_{i=1}^m (\lambda_i - \lambda_i^*) d_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m (\lambda_i - \lambda_i^*) (\lambda_j - \lambda_j^*) \mathbf{x}_i \cdot \mathbf{x}_j - \varepsilon \sum_{i=1}^m (\lambda_i + \lambda_i^*), \quad (4.60)$$

subject to the constraints

$$\begin{cases} \sum_{i=1}^m (\lambda_i - \lambda_i^*) = 0, \end{cases} \quad (4.61)$$

$$\begin{cases} 0 \leq \lambda_i \leq C \end{cases} \quad \text{for } i=1, \dots, m. \quad (4.62)$$

Similarly as discussed in §4.1, the formulation of the dual problem reduces the dependency of Q only on dot products of input vectors $\mathbf{x}_i \cdot \mathbf{x}_j$, targets d_i and Lagrange multipliers $\boldsymbol{\lambda}$, $\boldsymbol{\lambda}^*$.

The regression hyperplane can be expressed as well using (4.56):

$$f(\mathbf{x}) = \mathbf{w}_0 \cdot \mathbf{x} + b_0 = \sum_{i=1}^m (\lambda_i - \lambda_i^*) \mathbf{x}_i \cdot \mathbf{x} + b_0, \quad (4.63)$$

which is called the *support vector expansion*. Similarly to (4.38), the regression line is expressed entirely on training samples dot products (\mathbf{x}_i, d_i) . By this expansion, one can neglect the knowledge of \mathbf{w}_0 . To find the corresponding value of b_0 , one needs to apply conditions provided by the Kuhn-Tucker theorem to constraints (4.51), (4.52), (4.53) and (4.54), and associated multipliers:

$$\lambda_{0,i} (\varepsilon + \xi_i + d_i - y_i) = 0, \quad (4.64)$$

$$\lambda_{0,i}^* (\varepsilon + \xi_i^* - d_i + y_i) = 0, \quad (4.65)$$

$$\gamma_{0,i} \xi_i = 0 \implies (C - \lambda_{0,i}) \xi_i = 0, \quad (4.66)$$

$$\gamma_{0,i}^* \xi_i^* = 0 \implies (C - \lambda_{0,i}^*) \xi_i^* = 0. \quad (4.67)$$

These four equations allow to draw important conclusions. First of all, (4.66) and (4.67) allow to infer that every example (\mathbf{x}_i, d_i) for which $\lambda_{0,i} = \lambda_{0,i}^* = C$, are the only ones that can be such that $\xi_i > 0$ and $\xi_i^* > 0$; these slack variables correspond to points lying outside the ε -insensitive. Multiplying equation (4.64) by $\lambda_{0,i}^*$ and (4.65) by $\lambda_{0,i}$, and then adding the resulting equations, yields

$$\lambda_{0,i} \lambda_{0,i}^* (2\varepsilon + \xi_i + \xi_i^*) = 0. \quad (4.68)$$

The quantity in parenthesis is always bigger than zero since $\xi_i \geq 0$ and $\xi_i^* \geq 0$. This gives the condition $\lambda_{0,i} \lambda_{0,i}^* = 0$, from which it follows that there can never be a situation where the pair of Lagrange multipliers are both simultaneously non-zero. Examples (\mathbf{x}_i, d_i) for which the Lagrange multipliers are non vanishing define the support vectors, which also in the case of regression implies *sparseness* of the solution. From (4.66), it is possible to write that $\xi_i = 0$ as long as $\lambda_{0,i} \in (0, C)$ (the correspondent $\lambda_{0,i}^*$ will be zero according to the aforementioned condition). This will yield from (4.64):

$$\varepsilon + d_i - y_i = 0 \quad \text{for } 0 < \lambda_{0,i} < C. \quad (4.69)$$

Recalling that $y_i = \mathbf{w}_0 \cdot \mathbf{x}_i + b_0$, using this latter with condition (4.69) finally gives b_0 , under the condition of choosing an example $(\mathbf{x}^{(s)}, d^{(s)})$ such that (4.69) is satisfied:

$$b_0 = \varepsilon + d^{(s)} - \mathbf{w}_0 \cdot \mathbf{x}^{(s)}. \quad (4.70)$$

As a concluding remark, all methods here discussed for the linear regression are extendable to non-linear fitting function using the kernel trick, which would generalize equations (4.60) and (4.63) substituting the dot products with kernel ones.

4.2 Feature Extraction

Heart rate predictions were performed only on vowels segments, leaving out text-readings and free-speeches (see the description of the dataset in §5.1), as there is still no well-established literature investigating the effects of heart pumping on speech. The said segments were transformed in training instances using audEERING's benchmark product: openSMILE. This is an open source toolkit distributed by audEERING and it has become a standard in acoustic feature extraction for real time speech and music analysis [14, 15, 16]. Its main strengths are summarized as follows:

- **incremental processing**, where data from an arbitrary input stream (file, sound card, etc.) is pushed through the processing chain sample by sample and frame by frame;
- **ring-buffer memory** for features requiring temporal context and/or buffering, and for reusability of data, i.e. to avoid duplicate computation of data used by multiple feature extractors such as FFT spectra.
- **fast and lightweight algorithms** carefully implemented in C/C++, with no third-party dependencies for the core functionality;
- **modular architecture** which allows for arbitrary feature combination and easy addition of new feature extractor components by the community via a well structured API and a run-time plug-in interface;

- **configuration file**, defining the configuration of feature extractor parameters and component connections.

The openSMILE toolkit is capable to extract **low-level descriptors** (LLDs) and applying various filters, functionals and transformation to these [15]. An acoustic LLD is defined as a parameter computed from a short time frame of an audio signal. The length of the frame should be chosen to a) ensure quasi-stationarity with respect to the LLD of interest and b) ensure that the frame contain enough data to compute the LLD [14]. LLDs can be extracted directly in time domain, as well as in frequency domain. Therefore, given a signal $x[n]$ (which is in general an audio signal, even if theoretically openSMILE is applicable also to other kinds of signals) and dividing it in frames, it is possible to assign an LLD (e.g. RMS energy, short-time spectrum, MFCC coefficients, etc.) frame by frame. The higher the number of LLDs associated to $x[n]$, the richer is its description, eventually underlying properties of $x[n]$ that are not easily seen in time or frequency domain. A functional \mathfrak{F} maps a series of values $x[n]$ to a single value $X_{\mathfrak{F}}$. Indeed, to handle segments with variable length and get rid of the dependency of the feature vector dimensionality on the segment length, statistical functionals can be applied to the time-series of LLDs. Thus, the result is independent of the length of the input. Examples of commonly used functionals are the arithmetic mean, standard deviation, maximum value and minimum value. [14]. For example, given three different LLDs and applying to them the arithmetic mean functional will produce the first three entries of the feature vector associated to $x[n]$. The exhaustive number of algorithms to compute LLDs and functional is what allows to obtain non-redundant large-space feature vectors. Moreover, due to the modular architecture, it is possible to apply any implemented processing algorithm to any time series, i.e. the Mel-band filter-bank could be applied as a functional to any feature contour passed as data source to openSMILE. This gives researchers an efficient and customizable tool to generate millions of novel features without adding a single line of C++ code. These characteristics make openSMILE a comprehensive, cross-domain feature set extractor, and thus it has been considered the best choice for feature extraction also for this study. Each vowel speech segments were converted into 6373 dimensional feature vectors, using the **ComParE 2016** feature set. In this set, the LLDs (reported in detail in Table 4.2) are 65 in total, divided in two groups, to which 39 functionals are applied.

LLD	Functionals
Group A: (59)	Arithmetic or positive arithmetic mean
Loudness	Root-quadratic mean flatness
Modulation loudness	Standard deviation, skewness, kurtosis
RMS energy, ZCR	Quartiles 1–3
RASTA auditory bands 1–26	Inter-quartile ranges 1–2, 2–3, 1–3
MFCC 1–14	99 th and 1 st percentile, range of these
Energy 250–650 Hz	Revalite position of the max. and min. value
Energy 1–4 kHz	Range (maximum to minimum value)
Spectral RoP .25, .50, .75, .90	Linear regression slope a , offset b
Spectral flux, entropy, variance	Linear regression quadratic error
Spectral skewness and kurtosis	Quadratic regression coeff. a , b , c
Spectral slope	Quadratic regression quadratic error
Spectral harmonicity	Temporal centroid
Spectral sharpness (auditory)	Peak mean value and dist. to arithm. mean
Spectral centroid (linear)	Mean and std. dev. of peak to peak distances
Group B: (6)	Peak and valley range, peak-valley-peak slopes mean and std. dev.
F_0 via SHS, Prob. of voicing	Segment length mean, min., max., std. dev.
Jitter (local and delta),	Up-level time 25%, 50%, 75%, 90%
Shimmer	Rise time, left curvature time
log HNR (time domain)	Linear prediction gain and coefficients 1–5

Table 4.2: LLDs and functionals for ComParE 2016.

4.3 Classification and Regression Models

Due to the high dimension of the feature space and the relatively small amount of training instances, SVM has been chosen for modeling [44, 48]. As mentioned in §1.4, predictions from data will consist in classification and regression. In both cases, the feature set described in the previous section is used. Labeling depends on the problem under consideration (classification/regression), although labels are substantially based on the heart rate measurement described in

§3.3.2. For the first case, the labels are binary and they are produced setting a threshold to 100 BPM on the value produced by the heartbeat detection algorithm on the portion of ECG related to the vowel speech segment. For example, a value of 85 BPM associated to the segment will be assigned to the "Low" class, whereas a value of 115 BPM will be assigned to the "High" class. Then, such feature matrix (i.e. feature vectors plus labels organized in a data matrix) will feed a Support Vector Classifier (SVC). For the second case, the model must be presented with labels varying inside a certain interval, in order to predict a continuous output. Hence, recalling the previous example, the label assigned will be the BPM value obtained on the related segment, without further processing. The feature matrix will be then feed to a Support Vector Regressor (SVR). As pre-processing steps, data shuffling, feature normalization and feature selection were applied. In particular, for feature selection, two different reduction methods have been tested: **Univariate Feature Selection** (UFS), a standard feature selection procedure used to select a subset of features with the higher strength of relationship with the response variable, and **Weight Feature Selection** (WFS), a technique that involves a linear SVM trained on the whole feature set and thereby selecting the first n features associated with first n highest elements, taken with absolute value, of \mathbf{w}_o [9, 11]. For both classification and regression, an extensive grid search has been conducted to optimally tune SVMs' hyperparameters. Both model selection (via grid search) and evaluation were done using k -fold cross validation, one of the most widely used resampling methods [25, 26].

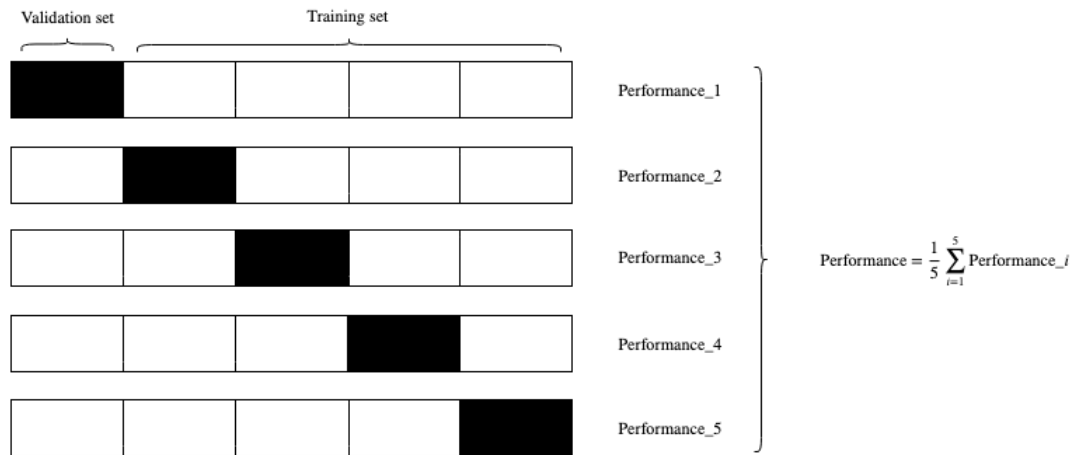


Figure 4.9: k -fold cross-validation.

The basic idea is that every sample is valuable both for training and testing. Holdout cross-validation methods always retain a part of the dataset (called the cross-validation set) to evaluate performances, without taking part to training. In k -fold cross-validation, a dataset is divided in a number of folds (Figure 4.9) containing all the same amount of training samples: every fold is used time by time as validation set, being the remaining $k - 1$ folds used as training set. When the loop has run through all folds, the performance is evaluated as the mean of the k performances obtained during the process. As already mentioned, this method applies both for optimal parameters estimation and model performance evaluation. Indeed, "performance" refers to a metric used throughout the process. For tuning the hyper-parameters, tuples containing every possible combinations from the parameter grid are tested with a suitable metric (for example, accuracy for tuning classifiers or MSE for regressors): performance is obtained then as the mean value of the chosen metric computed with k -fold. The tuple leading to the highest performance determines in this way the optimal hyperparameters for the model at hand. Model selection and evaluation by k -fold cross validation is a statistically robust approach when the amount of data is restricted [42], and for this reason it was chosen also for this study (with $k = 10$).

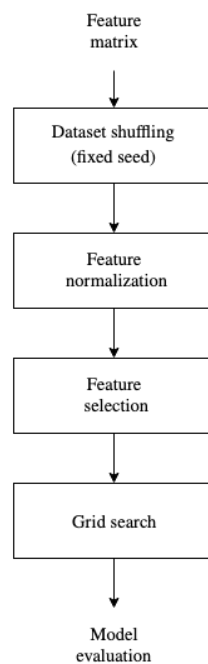


Figure 4.10: Modeling pipeline.

Chapter 5

Experimental Results and Discussion

In previous chapters, the implementation of the acquisition system managed by the BioMiner application have been explained and discussed in detail. Likewise, computational methods were introduced to realize audio-based heart rate prediction. In this chapter, data collection is described, as well as the content and organization of the dataset itself. It is shown how labeling is done through the data sanity-check feature implemented in the BioMiner’s analysis mode, assessing also the accuracy of the heartbeat detection algorithm. The dataset is thus compared to the MBC, outlining weaknesses and strengths of the new data with respect to the already existing one. Then, SVM models are trained on vowel speech segments extracted from the dataset, using the processing steps reported in §4.3 in many different conditions, with several metrics employed evaluating case by case. The chapter is concluded by discussing the goodness and limitations of this study.

5.1 Data Collection

Data collection was led internally to audeERING’s seat, Gilching office, Germany. As discussed throughout previous chapters, data was acquired in a closed, indoor environment, using

the experimental setup described so far. A single session consisted of the participant sitting on the fitness cycle and following the instructions provided by the application. All participants were at the time employed or directly involved with the company. They were aware of the modalities and finalities of the study by means of a detailed **participant information sheet**, containing in plain: a general overview of the data collection procedure, aims of the study, voluntariness of participation, risks and modalities of treatment of personal information. This latter point was particularly important. In fact, the collected data (audEERING retains its full ownership) is meant for research purposes or research challenges, following a strict privacy policy, where the confidentiality, storing and protection of data are paramount aspects. As mentioned in §3.1.1, participant’s personal information is not permanently kept and a one-way hash code is produced instead. This code, in addition to providing anonymity of every dataset’s record, was also given to the participant at the moment they sign the **consent form**, where they accepted for the treatment of data collected from them for the above-mentioned purposes. In exchange, they received their unique hash code, to be conserved for future modification or cancellation of their records.

5.1.1 Experimental Protocol

After reading the information sheet and signing the consent form, every participant had to stick to the same experimental pipeline in order to provide repeatable, comparative and cumulative data. After a quick review of the motivation, risks and purposes of the study, the participant was connected with electrodes as shown in Figure 3.11; at the same time, they had to head-worn headsets (Figure 2.8) and seat on the fitness cycle. Literature has shown that heart rate and breath rate can be determined by embedded cameras on both laptops [39] and smartphones [49] with great accuracy: the possibility of video-based heart rate estimation indeed induced to add also video recording to the experiments. However, to respect confidentiality of sensitive information about the participant, only those who explicitly accepted to acquire also video data from the experiments were video-recorded. Therefore, for those agreeing, the first step was to light up camera lights and turn on the GoPro, which was positioned on the cycle handlebars as in Figure 5.1. Afterwards, the participant had to provide a set of personal and

generic information, as described in §3.1.1.



Figure 5.1: Fitness cycle detail: GoPro camera.

After that, a one minute long ECG test (*ECG test* button, see Figure 3.1) was done to check ECG signal quality, and eventually make troubleshooting in case of excessive noise, connection problems or hardware malfunctioning. The session then started. Each session was made up by seven consecutive tasks, to be fulfilled during as many trials (or experiments). These were to be carried at rest (i.e. no cycling) or in exercise condition, which was achieved by cycling on the fitness cycle. The level of physical exercise was controlled by a predefined threshold based on

BPMs displayed in real-time during each single experiment. Tasks were completed in this order:

1. sustained phonation of the entire set of vowels (/a/, /e/, /i/, /o/, /u/) at rest, each uttered at least twice;
2. reading of a standard phonetic text at rest, namely: *Nordwind und Sonne* for German natives, *The North Wind and the Sun* for the others;
3. sustained phonation of the entire set of vowels (/a/, /e/, /i/, /o/, /u/), each uttered at least twice and each preceded every time by physical exercise up to 100 BPM;
4. sustained phonation of the entire set of vowels (/a/, /e/, /i/, /o/, /u/), each uttered at least twice and each preceded every time by physical exercise up to 130 BPM;
5. exercise up to 140 BPM, prolonged for 10 s;
6. reading of the aforementioned phonetic text while recovering from exercise;
7. free-speech part consisting of a final interview made of five generic questions.

This pipeline defines the **experimental protocol** and is implemented following the instructions to write the configuration file, provided in §3.2.2, allowing every participant to efficiently follow the progress of the session thanks to the BioMiner application. Every step is assigned with a label, corresponding to the `task_title` attribute of the configuration file: "Vowels" for steps 1, 3, and 4; "Reading" for steps 2 and 6; "Exercise" for step 5; and "Interview" for step 7. With this protocol, annotated audio segments contained in the dataset clearly define three distinct groups: sustained vowels segments, reading text segments, and free-speech segments. Thanks to physical exercise, these segments are indeed provided with heart rate labels in a wide range of values. Potentially though, many other classes could be retrieved, such as spontaneous arousal of laughter, emotional voicing, fatigue and breathing sounds due to cycling. Yet, these other types are not explicitly declared in the experimental protocol, so they cannot be annotated in the dataset with the methods described in §3.1.2, and must be thus segmented and annotated either manually or with automatic annotation tools (for example NOVA, an annotation tool based on *cooperative machine learning* to find similar segments on the basis of a small number of pre-annotated training sample of the same kind [59]).

5.1.2 Dataset Description

The organization and contents of the dataset are represented in Figure 5.2. Every folder is called a *node* in the directory hierarchy of the dataset, which is stacked in three levels. Immediately after the dataset's root, there is the *participant level*. This level contains all nodes related to every single participant that took part to the study, and every node contains all possible sessions that the participant had fulfilled in different occasions. Ideally, every participant is meant to participate only once to the data collection; yet, should the participant need to repeat it again in a different date, the new record will be saved as another element of the same participant's node.

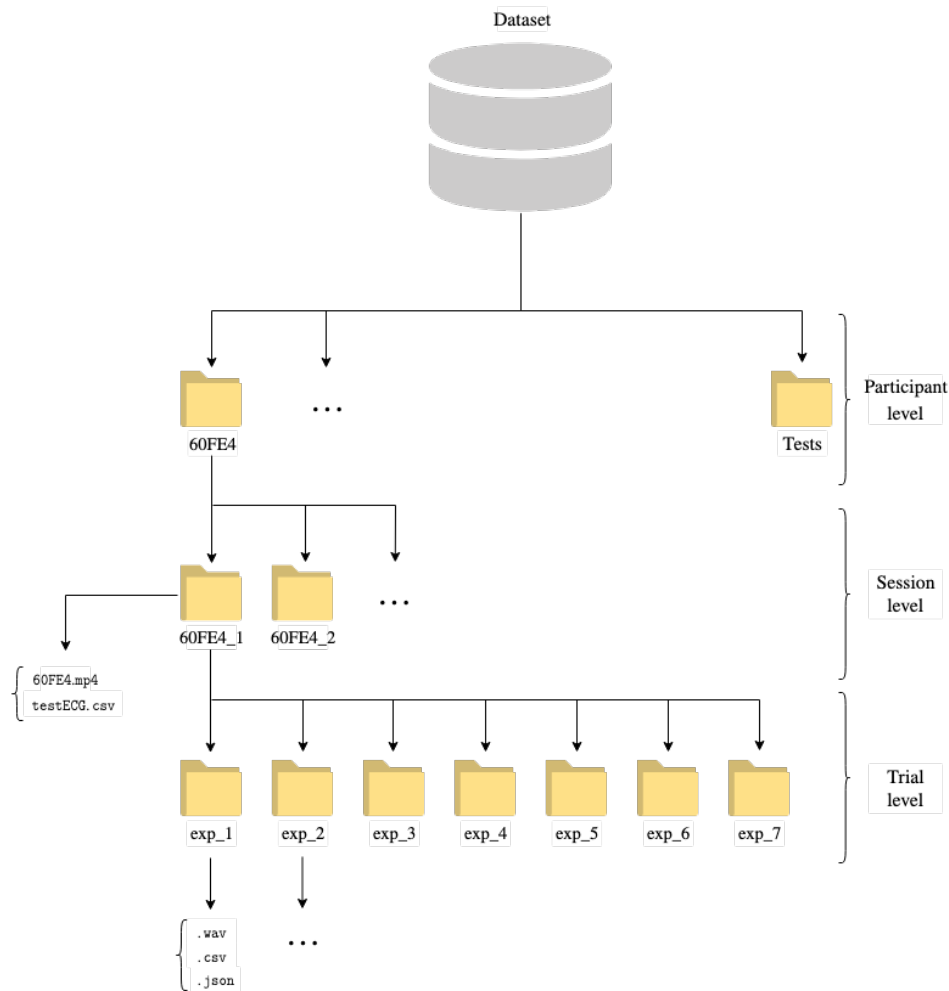


Figure 5.2: Directory tree of the dataset.

To be noted, every node is named after the first five digits of the participant's hash code, thus hiding their identity. Moreover, all those takes that were not meant to follow the experimental protocol, but just used for testing purposes, are saved in the *Test* folder, avoiding to delete permanently data which is however of possible use in future. Then, there is the *session level*: any record of the dataset is contained in a node of this level, i.e. any successfully completed session is contained in such folders, even those belonging to the same participant. Any node of this level contains also the video track and the *.csv* file related to the one-minute ECG test done before starting the session. The lowest level is the *trial level*, which contains every single trial compounding a session. Therefore, every node at session level contains 7 elements, as many as the steps of the experimental protocol. In turn, each node at trial level contains files associated to the recording between the start and end beeps. These are: a *.wav* file containing the audio recording of the related trial (beeps included), a *.csv* file including the ECG signal with the formatting described in §2.2 and a *.json* file, which contains all metadata and annotations related to the related trial. This latter one is particularly important because it contains all the information necessary to retrieve any particular segment and assign it with appropriate labels.

```
{
  "trial": {
    "id": "v2:60fe4e2ad2",
    "date": "18/3/2019",
    "participant": {
      "age": 25,
      "weight": 85,
      "height": 175,
      "gender": "M",
      "country": "Italy",
      "optionals": {
        "lifestyle": "sedendary",
        "habits": {
          "smoking": false,
          "alcohol": "rare",
          "caffeine": "regular",
          "drugs": false,
```

```
        "stress": "regular"
      },
      "speech_related": {
        "cold": false,
        "stuttering": false,
        "swallowing": false,
        "toothache": false
      },
      "previous_diseases": false
    }
  },
  "equipment": {
    "board": "Arduino UNO",
    "shield": "Olimex SHIELD-EKG-EMG",
    "mic": "Sennheiser PC 8 USB Headset"
  },
  "task": {
    "duration": 179.708,
    "synchronization": 99.922,
    "name": "Vowels",
    "subtasks": [
      {
        "label": "A",
        "start_time": 6.908,
        "end_time": 16.845,
        "comment": "nd",
        "token": "V",
        "mean_BPM": 79.1
      },
      {
        "label": "A",
        "start_time": 29.057,
        "end_time": 38.17,
        "comment": "nd",
        "token": "V",
        "mean_BPM": 76.889
      }
    ]
  },
}
```

```
{
  "label": "E",
  "start_time": 47.66,
  "end_time": 58.518,
  "comment": "nd",
  "token": "V",
  "mean_BPM": 75.364
},
{
  "label": "E",
  "start_time": 66.568,
  "end_time": 74.95,
  "comment": "nd",
  "token": "V",
  "mean_BPM": 69.375
},
{
  "label": "I",
  "start_time": 81.05,
  "end_time": 90.595,
  "comment": "nd",
  "token": "V",
  "mean_BPM": 74.9
},
{
  "label": "I",
  "start_time": 98.284,
  "end_time": 107.397,
  "comment": "nd",
  "token": "V",
  "mean_BPM": 73.0
},
{
  "label": "U",
  "start_time": 112.383,
  "end_time": 122.085,
  "comment": "nd",
```

```
    "token": "V",
    "mean_BPM": 73.8
  },
  {
    "label": "U",
    "start_time": 126.213,
    "end_time": 129.5,
    "comment": "nd",
    "token": "X",
    "mean_BPM": 73.333
  },
  {
    "label": "U",
    "start_time": 134.283,
    "end_time": 143.859,
    "comment": "nd",
    "token": "V",
    "mean_BPM": 78.7
  },
  {
    "label": "0",
    "start_time": 149.234,
    "end_time": 158.404,
    "comment": "nd",
    "token": "V",
    "mean_BPM": 72.333
  },
  {
    "label": "0",
    "start_time": 163.376,
    "end_time": 173.539,
    "comment": "nd",
    "token": "V",
    "mean_BPM": 73.545
  }
],
"ecg": {
```

```
        "beats": [
            0.76,
            1.46,
            2.16,
            2.85,
            3.56,
            ...
        ],
        "accuracy": 100.0
    }
}
```

As discussed in §3.2.1, at the end of every trial, `writeMetadata()` converts the *metadata buffer* in such a `.json` file, structured as in the previous listing. In the example presented above, this one is related to the first trial of the experiment protocol. The root element is the **trial** JSON object: this one collects all possible information about the participant, equipment, task and subtasks related to the correspondent trial. Its first fields contain all information provided in the input forms described in §3.1.1, exception made for the set of confidential information (first name, last name, place of birth, birthdate). The participant code is also kept here: in fact, a record of this dataset must be immediately retrieved should the participant appeal to the right of modification or deletion by providing their own participant code (as stipulated by signing the consent form). Then, two of its important attributes are **equipment** and **task**. The first one (**equipment**) describes the equipment used to acquire data. To be noted, this part, together with participant's information, remains constant over consecutive trials, because neither the participant nor the equipment can be changed in between. Together, they constitute the effective block of metadata related to a recording session. The second one (**task**) is devoted instead to collect annotations about audio and ECG data. Hence, this part differs for each node at trial level. As appears clear from the example, it has three attributes generally describing the task of the current trial: **name**, which indicates the type of task being done during this trial (in this case the possibilities are: "Vowels", "Reading", "Exercise", "Interview"), **duration**, which indicates the duration of the

task measured in seconds, and **synchronization**, a factor measuring the mismatch η between the task duration provided by Arduino's timestamps and the task duration calculated using the beep detection algorithm described in §2.3.1, calculated as

$$\eta = \frac{\Delta t_{\text{arduino}}}{\Delta t_{\text{beeps}}} \cdot 100. \quad (5.1)$$

To be noted, $\Delta t_{\text{arduino}}$ is assigned to **duration**. Then, the other fields contain information about audio/ECG segments, organized in subtask JSON objects, ordered in a list. If a task does not comprehend subtasks, this list is empty. The first five attributes per subtask object contain values accumulated in the metadata buffer during the ongoing trial. They are: **label**, containing the label describing the subtask, **start_time** and **end_time**, containing respectively the start and end timestamps of the segment related to the subtask, **comment**, containing a short description string of the subtask, and **token**, which allows only binary values indicating success (or not) of the subtask. The attribute **mean_BPM** is appended to each subtask object only after checking every trial node using the application's *analysis mode*. The procedure has been already mentioned in §3.1.2 as sanity-check of the dataset, and is necessary to correct manually erroneous R peak locations produced by the heartbeat detection algorithm, as shown in Figures 5.3, 5.4. The said procedure articulates in two steps: a) load a node from trial level using the analysis mode of the application. The heartbeat detection algorithm finds all R peak locations which are displayed on the screen as black dashed lines; b) inspect the whole ECG track and correct manually the erroneous R peak locations (as described in §3.1.2), then, save the corrected locations permanently. This last action will indeed add a new attribute to **task**, that is **ecg**, whose attributes are: the corrected R-peaks locations expressed in seconds (**beats**, truncated for representative purpose to the fifth element in the code listing above), and the accuracy of the algorithm applied to the ECG signal related to the loaded node (**accuracy**). At this moment, every subtask object's **mean_BPM** attribute is filled with the corresponding value. In fact, knowing now the corrected R peak locations and the start-end timestamps for every segment, it is easy to calculate BPM over that segment using (3.9). This last point covers a certain importance: in fact, only at this moment the target label to be used for model training is reliably produced.

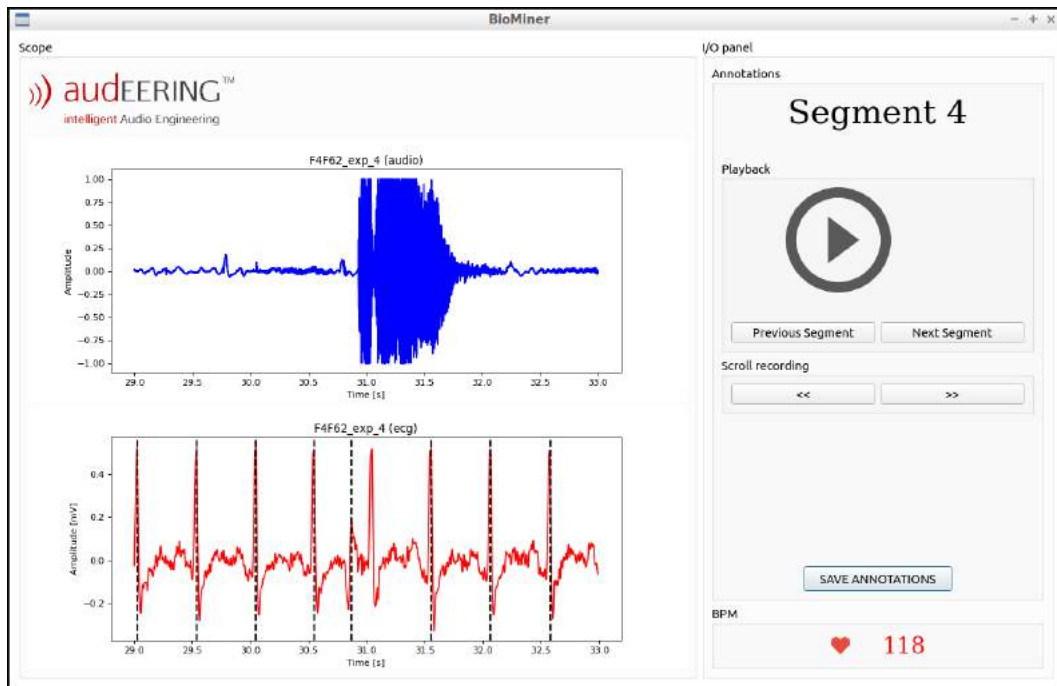


Figure 5.3: Example of a false positive (incorrectly detected peak) and a false negative (peak not detected at all).

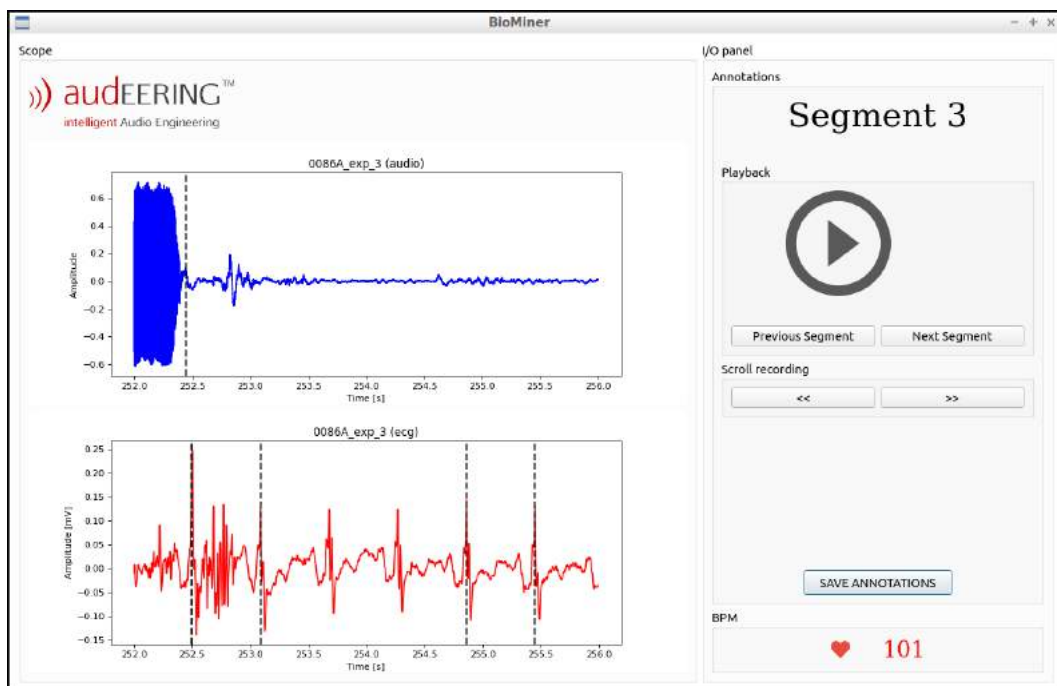


Figure 5.4: Example of two consecutive false negatives.

It must be observed that also an accuracy value is reported in the `.json` file. To explain the rationale behind the calculation of this index, some clarifications are needed. Theoretically, the output of the heartbeat detection algorithm can be:

- a detected peak where the peak is effectively present (true positive);
- a detected peak where the peak is, on the contrary, absent (false positive);
- a non-detected peak where the peak is effectively present (false negative);
- a non-detected peak where the peak is absent (true negative).

Although the algorithm described in §3.3 is a detection algorithm, it could be recast as a binary classification algorithm. In this viewpoint, each signal sample is assigned with a binary value, 1 for a detection, 0 for a non-detection. Then, the sanity-check procedure allows to inspect the output of the algorithm, and eventually correct it. For instance, every time a peak is falsely detected (like in Figure 5.3) the corresponding signal sample will be a false positive. Again, every time a peak is not detected (like in Figure 5.4), the corresponding will be a false negative, and so on. At the end of phase b) of sanity-checking of an ECG signal associated to the node loaded, the sum of true positives, true negatives, false positives and false negatives are indicated respectively as TP , TN , FP , FN . Yet, with this reasoning, TN would count more than the other three, because QRS complexes are isolated events as opposed to the baseline. Moreover, the higher the sampling rate of the signal, the higher grows TN . A first proposal for an evaluation index, neglecting true negatives, would be precision:

$$p = \frac{TP}{TP + FP}. \quad (5.2)$$

Yet, precision does not take in account false negatives either, and thus non-detected peaks would not be used to penalize appropriately the algorithm. To overcome this, a suitable metric considering only TP , FP and FN is the F_1 score, calculated as

$$F_1 = \frac{2TP}{2TP + FP + FN}. \quad (5.3)$$

With the F_1 score, the overall accuracy of the algorithm on the dataset is higher than 99%.

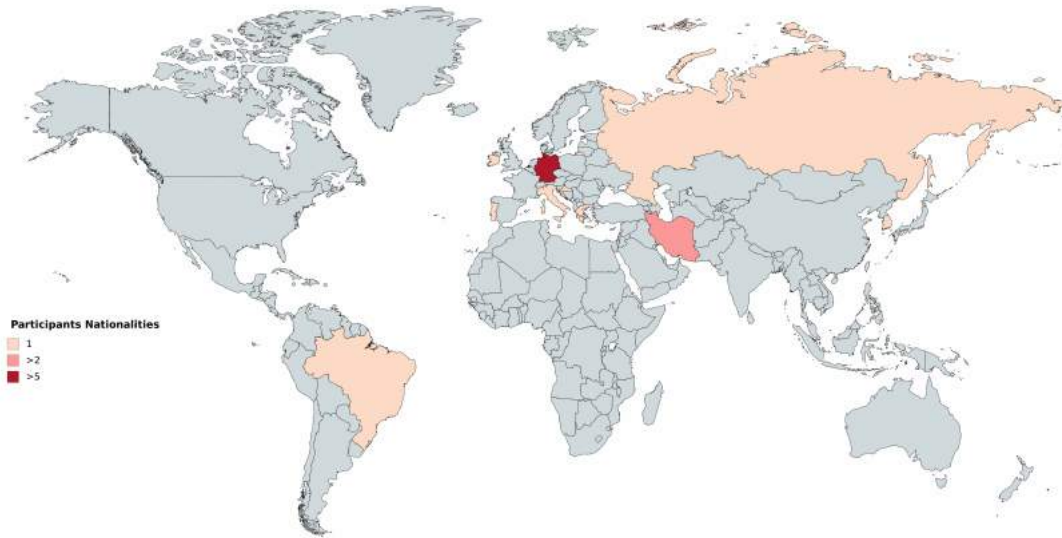


Figure 5.5: Germany: 20 participants (in red); Iran: 3 participants (in pink); Brazil, Croatia, Greece, Ireland, Italy, Portugal, Russia, South Korea: totalling 8 participants (in pale pink).

The dataset has been recorded by an international group of 31 participants, 21 males and 10 females. Two thirds circa are German natives; the rest are from Europe, Asia and South America, as depicted in Figure 5.5. Among the participants, 26 gave consent also for video recording. The average participant age is 30.839 years, with the younger participant aged 20 and the older one 43, meaning that all participants were adults. As already said, also height and weight were saved for every participant, allowing to calculate the body mass index (BMI), a biometric index to indicate individual's fatness [36]. The average BMI is 23.018 kg/m^2 , with a minimum of 18.726 kg/m^2 and a maximum of 32.846 kg/m^2 . This means that every participant statistically was in their normal weight condition, suggesting that they were not excessively fatigued during tasks involving physical exercise. Anyway, six participants had their $\text{BMI} > 25 \text{ kg/m}^2$, meaning that roughly one fifth of the participants was over-weight during the period of data collection. More than half of the participants (18) declared to practice sport on a regular basis (at least twice a week on average); the others (13) had sedentary habits. Other interesting statistics regard participant's habits, on the basis on the levels discussed in §3.1.1 ("None", "Rare", "Moderate", "Regular" and "Intensive"). Most of the participants (27) were non-smokers, the rest being: 2 rare smokers, 1 moderate, and 1 regular. Alcohol consumption:

11 none, 8 rare, 6 moderate, 6 regular. Caffeine consumption: 11 none, 3 rare, 4 moderate, 10 regular, 3 intensive. Stress, intended as susceptibility to psychological pressure: 12 none, 7 rare, 6 moderate, 5 regular, 1 intensive. All participant were not under assumption or deliberate usage of drugs except one, with rare frequency. The majority of participants were in normal medical condition, with one participant declaring to suffer from a mild *hypotension*. The only significant disease was encountered in a participant with a diagnosed *mitral prolapse*. Since the physical exercise was involved, it is noteworthy also to report that there was one participant recovering from foot injury. To conclude with, box plots are reported for steps 1, 3, and 4 of the experimental protocol, since the only segments used as training instances are those from "Vowels" tasks. To every participant is assigned their box plot for each of these steps. As shown in Figure 5.6, for task 1 boxes and whiskers are confined approximately between 60 and 100 BPM, which is the normal heart rate range for adults [24]. Nevertheless, it can be noted a certain dispersion of boxes, many with medians above 80 BPM and whiskers longer than the average. This suggest that some of the participants had already an excited state when they started with the task 1. Most of the participant were able to fulfill the tasks without varying that much their heart rate. This latter fact translates in small interquartile range (IQR) and short whiskers.

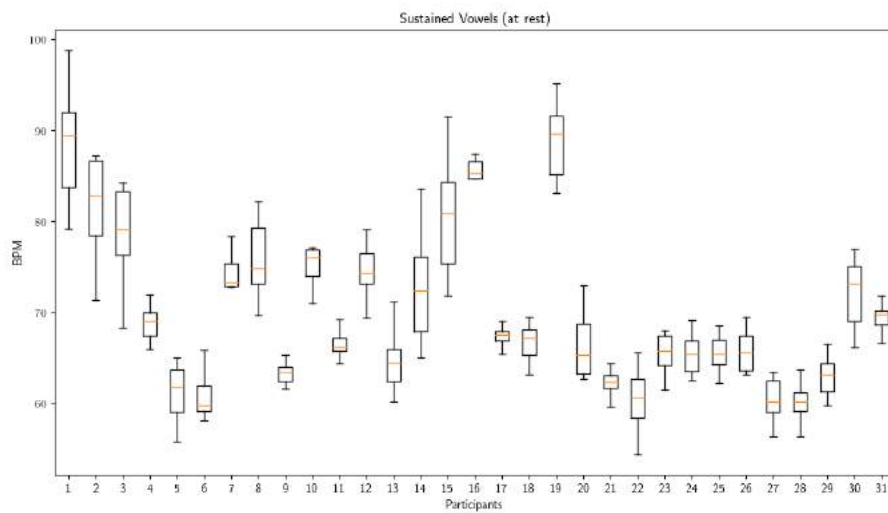


Figure 5.6: Box Plot for Sustained Vowels (at rest).

For task 3, the dispersion is reduced and heart rates are approximately confined in the range between 80 and 120 BPM, with IQRs generally larger than the previous case. To be noted, one participant was completely off this trend and did not increase BPMs even near the threshold.

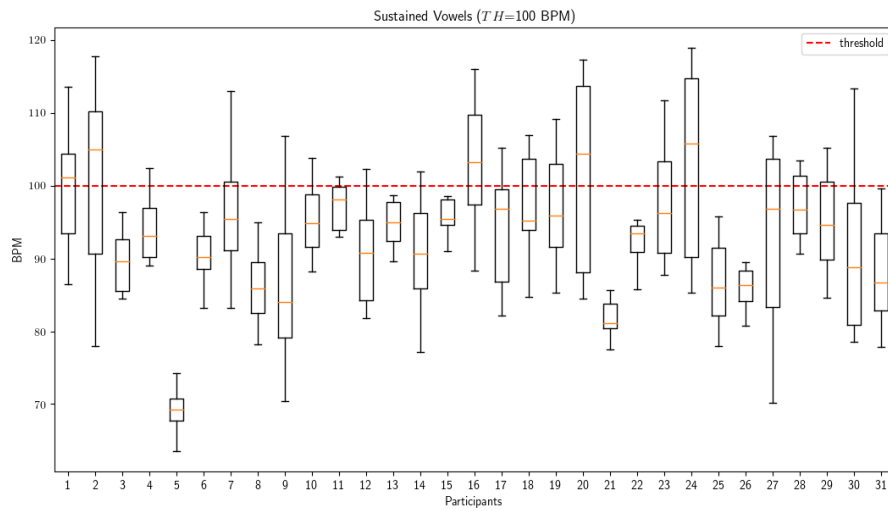


Figure 5.7: Box Plot for Sustained Vowel at 100 BPM.

For task 4, whiskers appear greater compared to task 3, with medians closer to threshold.

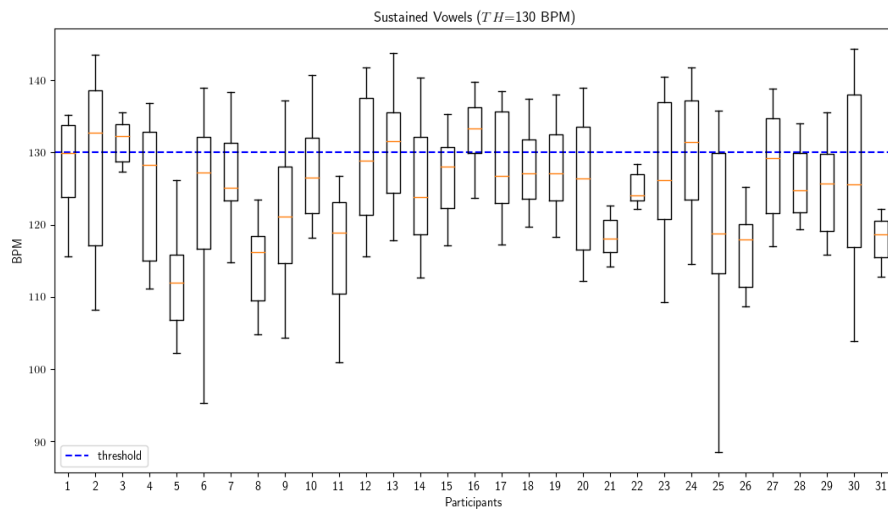


Figure 5.8: Box Plot for Sustained Vowel at 130 BPM.

5.1.3 Comparison with the MBC

The obtained dataset is compounded by the following annotated segments: 1003 vowels, 61 text readings and 158 free-speeches. Differently from the MBC, where the segments were cut and stored as new files, along with their HR and SC label values enclosed in the related segments' filenames, here the annotations are taken separately from the audio files. In fact, instead of cutting and storing segments in new files (increasing storing space), only original audio track files are kept, with any other metadata or annotation kept apart from it in the `.json` file of the segments' corresponding nodes. As widely discussed in the previous section, this file contains also detailed information about participants, differently from the MBC, which retains only HR and SC values as additional metadata to the audio files. With these precautions, there are three main advantages over MBC:

- possibility of future re-use of original recordings for different purposes (audio segmentation and annotation into new classes, e.g. laughter segments);
- increased perspectives in terms of correlation and inference thanks to the greater variety of available metadata;
- data redundancy avoided and storage space reduced.

Besides that, another improvement to the MBC is the *multimodal data* approach. In fact, audio, ECG and video were acquired at the same time, while in the MBC only audio files are present, with all other information being condensed in segment file's titles. However, the MBC provided two separate audio tracks (close-talk and room microphones), whereas in the present dataset only the recording from headset's microphone is available. The introduction of supervised physical exercise using fitness cycle (instead of climbing a staircase up and down) allowed to obtain target labels in a wide range and with good reproducibility (as shown in box plots in Figures 5.6, 5.7, and 5.8). In fact, although the MBC's HR labels approximately cover the same range, the variance of such labels is not as high as for those of the present study: the number of participants reaching such high values is restricted to a few individuals, whereas the experiments for this study were designed to reach the highest BPM values every time a recording took place. Indeed, the use of ECG instead of pulseoximetry allowed more precise measurements of BPMs, overcoming the

problems arising from the recording device used for the MBC (see §1.4). Anyway, the downfall of this choice is the impossibility of measuring also skin conductance values, which are here missing. Differently from the MBC, where only /a/ sustained vowels were used for experiments, in this case the entire set of vowels was used for recording and the training part for pitch intonation is absent. Finally, this dataset lacks of annotated breathing periods (but provides free-speech segments instead), which could be however retrieved as most of the participants were breathing loudly after exercising.

5.2 Results

Both classification (SVC) and regression (SVR) models were trained and validated following the steps described in §4.3. All feature sizes, used to train different models, varied over the set $\{100, 250, 500, 750, 1000, 2000, 3000, \text{"all"}\}$, where "all" indicates the use of the entire openS-MILE feature vector without feature selection. Both linear and Radial Basis Function (RBF) kernels were used, as well as both two feature selection methods (UFS, WFS). With this setup, a total of 60 different models were trained and evaluated. For good reproducibility of the results, all these were obtained using Scikit-learn Python's library, which makes available both cross-validation grid search and SVM dual problem solving in a robust and computationally fast way, being based on the famous LIBSVM C library. Furthermore, the low number of hyperparameters for such models allowed to perform computations in an amenable amount of time.

5.2.1 Classification

With linear kernel, the list of hyperparameters consisted of the C complexity parameter only. The possible values were determined on increasing power of ten scale, namely $\{10^{-5}, 5 \cdot 10^{-5}, 10^{-4}, 5 \cdot 10^{-4}, 10^{-3}, 5 \cdot 10^{-3}, 10^{-2}, 5 \cdot 10^{-2}\}$. To be noted, this list has been chosen such that every value is smaller than 1. In fact with linear kernel, C should not be set higher than 1 if there is no clue that the dataset's classes are linearly separable in the input space X . On the other hand, for RBF kernel, the list of hyperparameters was: $\{10^{-2}, 5 \cdot 10^{-2}, 10^{-1}, 5 \cdot 10^{-1}, 1, 5, 10, 50, 100\}$ for C , and $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ for σ . Models were evaluated in terms of accuracy, precision and recall, using both UFS and WFS methods. Results are summarized in Tables 5.1, 5.2.

	UFS			WFS		
	Accuracy	Precision	Recall	Accuracy	Precision	Recall
100	0.781	0.751	0.693	0.811	0.778	0.751
250	0.795	0.763	0.722	0.848	0.821	0.802
500	0.794	0.771	0.707	0.855	0.848	0.79
750	0.789	0.756	0.717	0.854	0.836	0.802
1000	0.794	0.77	0.712	0.855	0.839	0.8
2000	0.796	0.772	0.715	0.846	0.829	0.79
3000	0.805	0.774	0.739	0.834	0.822	0.76
"all"	0.785	0.767	0.688	0.785	0.767	0.688

Table 5.1: Classification results for linear kernel.

	UFS			WFS		
	Accuracy	Precision	Recall	Accuracy	Precision	Recall
100	0.789	0.76	0.71	0.816	0.782	0.763
250	0.793	0.768	0.71	0.85	0.826	0.802
500	0.797	0.775	0.712	0.86	0.852	0.8
750	0.802	0.779	0.722	0.858	0.841	0.81
1000	0.802	0.773	0.732	0.857	0.842	0.802
2000	0.813	0.781	0.756	0.851	0.838	0.795
3000	0.803	0.775	0.732	0.83	0.818	0.759
"all"	0.798	0.767	0.732	0.798	0.767	0.732

Table 5.2: Classification results for RBF kernel.

For both linear and RBF kernel approaches, all three metrics used increase with the number of feature selected with UFS, exception made when the whole feature vector is considered. Conversely, using WFS, these latter reach their best values when the number of features used is in the interval $[250, 1000]$. It must be observed that the model used for WFS in this case is a linear SVC trained with the whole feature set. Confusion matrices for the best models (highlighted in green in Tables 5.2, 5.3) are reported in Figures 5.9a, 5.9b and 5.10a, 5.10b.

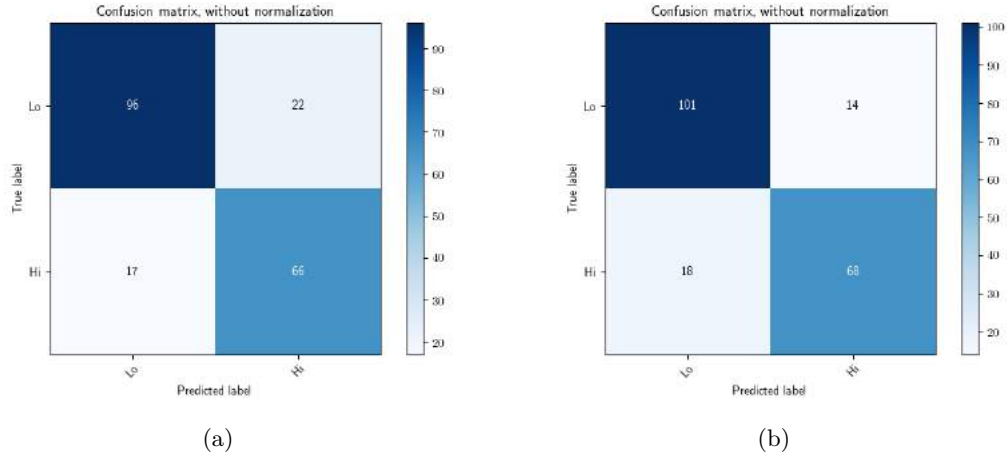


Figure 5.9: Confusion matrices for the best models with linear kernel, respectively: UFS ($n = 3000$, Figure 5.9a) and WFS ($n = 250$, Figure 5.9b).

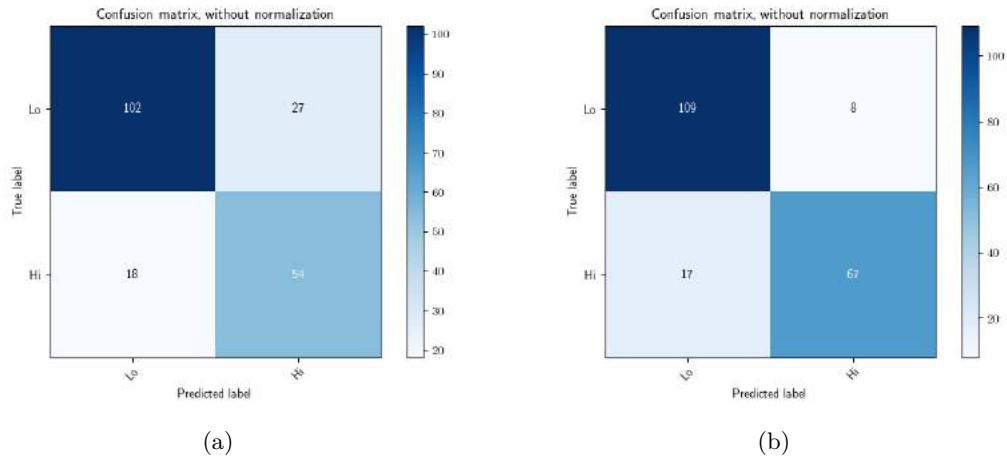


Figure 5.10: Confusion matrices for the best models with RBF kernel, respectively: UFS ($n = 2000$, Figure 5.10a) and WFS ($n = 500$, Figure 5.10b).

The absolute best classifier is the RBF kernel one with feature size (selected with WFS) of $n = 500$, with $C = 5$ and $\sigma = 10^{-4}$. As shown in Figure 5.10b, the true negative rate reaches some 93%, whereas the true positive rate is stuck to a 80% circa. This is possibly a downside of such audio-based classification: in fact, a common event, like resting heart rate (i.e. below 100 BPM), would be classified better than a way more rare event, like being nervous or under physical exercise (i.e. above 100 BPM), degrading model's sensitivity. To conclude with, diagrams showing hyperparameters vs. size of feature set are reported in semi-logarithmic plots

in Figure 5.11. It can be observed that the trend of C for linear kernel SVMs is in the order of magnitude of 10^{-3} , while for RBF ones it is 4 order of magnitude greater: this means that the decision surfaces of the second ones are highly complex, with the possibility of overfitting.

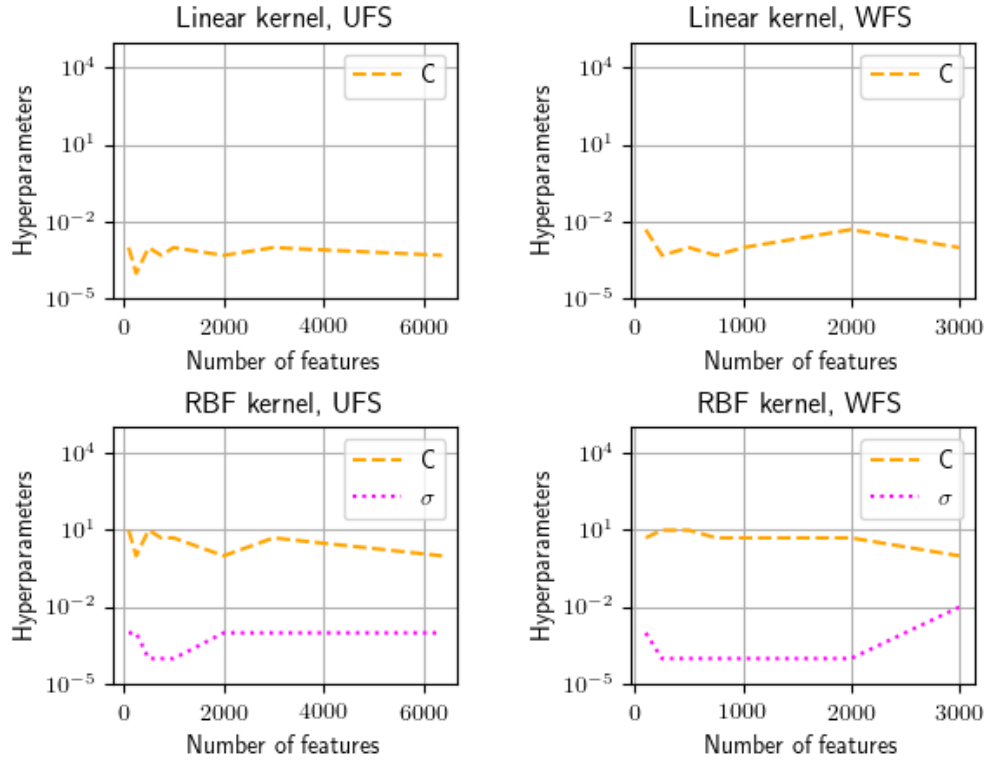


Figure 5.11: Hyperparameter curves for classification models.

5.2.2 Regression

Regression analysis follows quite similarly the steps of classification. In this case, also ε (defining the SVR insensitive tube, see §4.1.3) must be considered. This parameter has an effect on the smoothness of the SVM's response and it affects the number of support vectors, so both the complexity and the generalization capability of the model depend on its value. There is also some connection between observation noise in the training data and the value of ε . Setting it to the desired approximation accuracy does not guarantee the convergence of the model to that performance, reason why it is usually included in the grid search process. The hyperparameter values

considered in the grid are in this case: $\{10^{-5}, 5 \cdot 10^{-5}, 10^{-4}, 5 \cdot 10^{-4}, 10^{-3}, 5 \cdot 10^{-3}, 10^{-2}, 5 \cdot 10^{-2}\}$ (linear kernel) and $\{10^{-2}, 5 \cdot 10^{-2}, 10^{-1}, 5 \cdot 10^{-1}, 1, 5, 10, 50, 100\}$ (RBF kernel) for C , $\{10^{-2}, 5 \cdot 10^{-2}, 10^{-1}, 5 \cdot 10^{-1}, 1, 5, 10\}$ for ε , and, when using RBF kernel, $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ for σ . WFS method was implemented by a linear SVR, trained on the whole feature set. The metrics used are Mean Absolute Error (MAE) and Pearson's Correlation Coefficient (CC). In a similar fashion to the previous case, results are reported in Tables 5.3 and 5.4.

	UFS		WFS	
	Mean Absolute Error (MAE)	Correlation Coefficient (CC)	Mean Absolute Error (MAE)	Correlation Coefficient (CC)
100	15.282	0.622	12.715	0.747
250	13.785	0.694	11.633	0.785
500	13.726	0.69	10.744	0.814
750	13.599	0.699	10.793	0.812
1000	13.589	0.705	10.621	0.82
2000	13.528	0.715	10.805	0.818
3000	13.253	0.731	11.247	0.806
"all"	13.413	0.721	13.413	0.721

Table 5.3: Regression results for linear kernel.

	UFS		WFS	
	Mean Absolute Error (MAE)	Correlation Coefficient (CC)	Mean Absolute Error (MAE)	Correlation Coefficient (CC)
100	14.574	0.655	12.653	0.751
250	13.144	0.717	11.231	0.797
500	12.727	0.735	10.56	0.818
750	12.551	0.747	10.494	0.821
1000	12.241	0.759	10.244	0.829
2000	12.838	0.743	10.625	0.821
3000	12.57	0.76	11.066	0.809
"all"	12.783	0.751	12.783	0.751

Table 5.4: Regression results for RBF kernel.

With UFS, for linear kernel, MAE decreases and the CC increases with the size of the feature set. Instead, for RBF kernel, such monotonic trends for MAE and CC are not observed. With WFS, both for linear and RBF kernels, the decrease of MAE (and the increase of CC) is monotonic only up for $n = 1000$; for $n > 1000$, the trend reverses. Learning curves for the best regression models (highlighted in green in Tables 5.3, 5.4) are reported in Figures 5.12a, 5.12b (linear kernel) and 5.13a, 5.13b (RBF kernel).

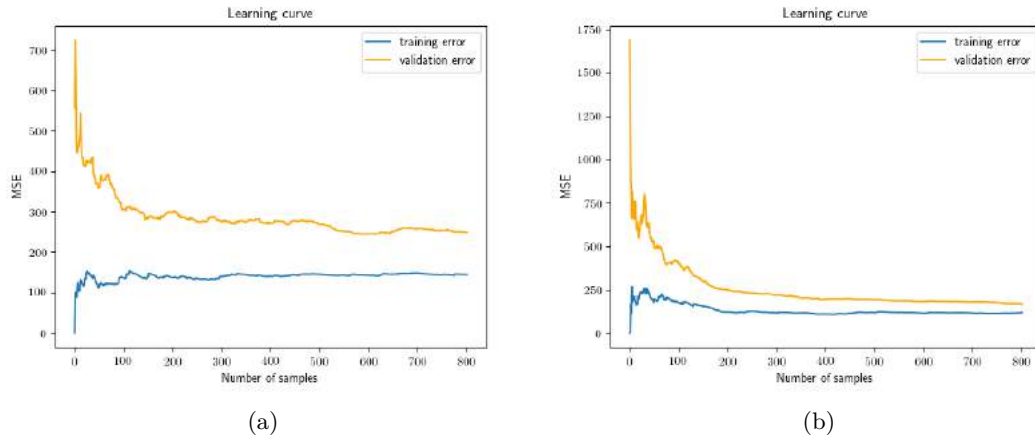


Figure 5.12: Learning curves for the best models with linear kernel, respectively: UFS ($n = 3000$, Figure 5.12a) and WFS ($n = 1000$, Figure 5.12b).

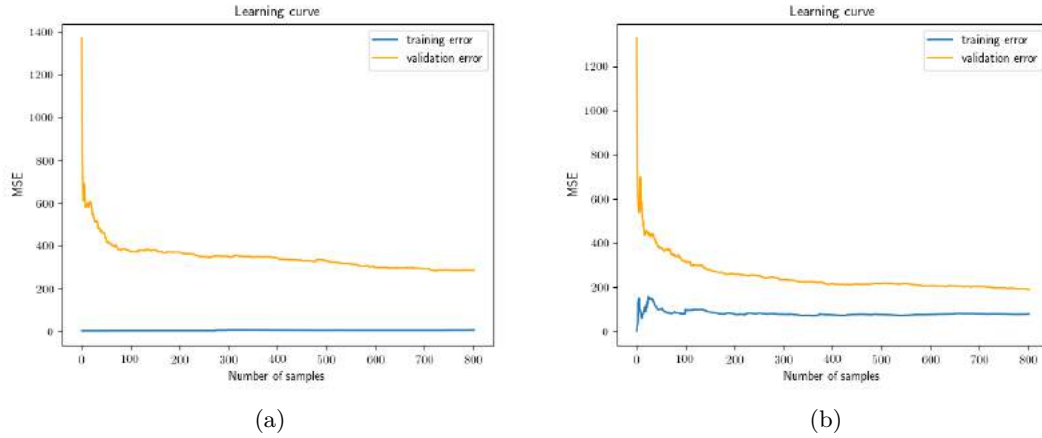


Figure 5.13: Learning curves for the best models with RBF kernel, respectively: UFS ($n = 1000$, Figure 5.13a) and WFS ($n = 1000$, Figure 5.13b).

Also in this case, the best model obtained is the one with RBF kernel using WFS method,

with $n = 1000$, $C = 100$, $\sigma = 10^{-4}$ and $\varepsilon = 0.1$. From learning curves, it is possible to observe that with UFS models, validation error is generally higher than WFS ones. This means that UFS models are affected with a bias problem. Since the learning curve gaps for UFS models are wider than WFS ones, this suggests that such models are affected with high variance too. In particular, in Figure 5.13a the training error is almost zero, suggesting that the related model overfits. For WFS, the gaps are smaller, suggesting that those models are less affected by variance than the UFS ones. Moreover, the curves tend to converge asymptotically, meaning that the add of new instance into the training set is unlikely getting better results. Even if the curves of Figure 5.12b does not correspond to the absolute best model, they represent the better bias-variance trade off. Analogously as for classification, hyperparameters vs. size of feature set are reported in Figure 5.14. As in the previous case, C tends to maintain a limited trend, while the trends of σ and ε tend to variate more, in a loosely correlated fashion.

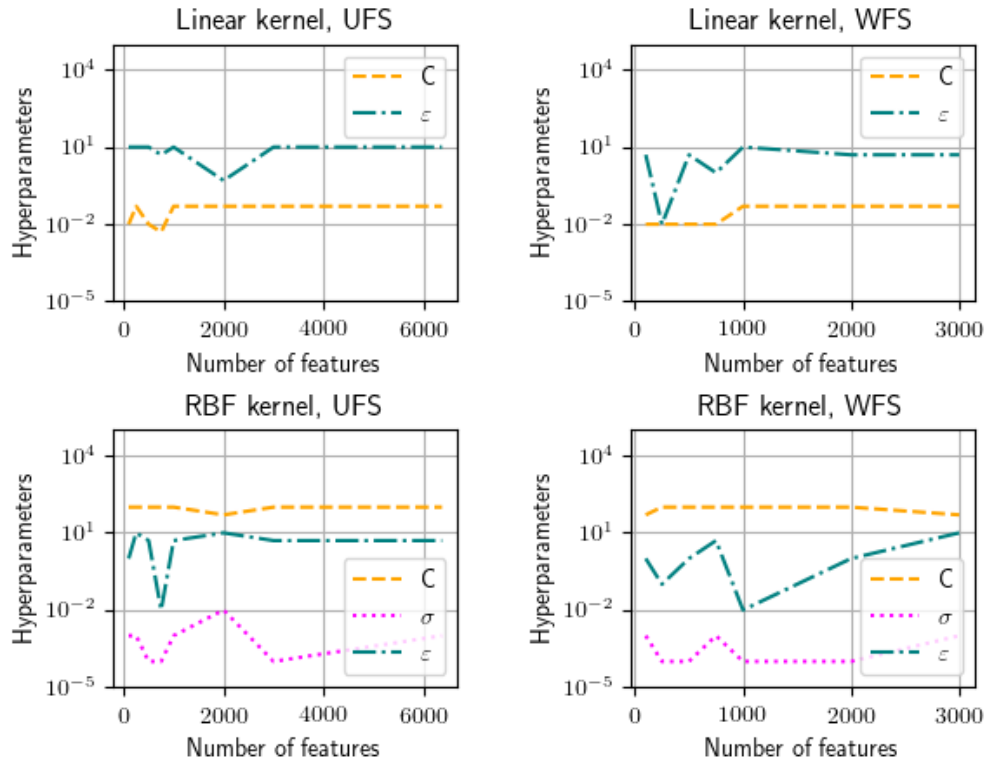


Figure 5.14: Hyperparameters curves for regression models.

5.3 Discussion

The meaning of the results of this study is bound to the methods and tools employed to obtain these. For regression analysis, it was possible to reach a $CC = 0.83$, indicating a strong correlation between the independent variable (heart rate) and the dependent one (acoustic features). The MAE was as low as 10.6 BPM. This result can be further interpreted comparing it to the range of variability of the BPM labels. Considering that targets and predictions are strictly positive, a possible robust, not biased, way to obtain an index of relative error is the Symmetric Mean Absolute Percentage Error (SMAPE) [1]. For the best SVR of Table 5.4, $SMAPE = 5.467\%$. Considering that 0% corresponds to perfect forecast and 100% to an indefinitely bad one, the obtained result is very promising for implementing reliable audio-based heart rate predictors. Also for classification, it was possible to reach an accuracy as high as 86%, with precision and recall above 80%. However, in addition to the already discussed concerns of not having a recall value as high as the other two, the main limitation of classification is that the models are build around a homogeneous group of adults, with an average age of 30 years. The average BPM over all experiments is 96 BPM circa, thereby justifying the choice of 100 BPM as a "Low/High" classes discriminant. However, it could be that another group of participants (e.g. elders) would reach a lower average BPM. This would influence the choice of the discriminant, and a new model would need to be trained. Therefore, the present classification model is too specific to the dataset, limiting performance to a wider population of subjects. Compared to the benchmark paper from Schuller et al. (where also the MBC is introduced) [48], the results are slightly improved, showing that similar outcomes can be obtained without the need of preferring one single type of vowel speech signal (e.g. the /a/ vowel) over the others, and without undergoing a training phase to adjust subject's pitch, availing spontaneous recognition from production of sustained vowels, without need of pitch intonation which can be sometimes difficult for untrained people. A limitation regarding both classification and regression resides on the fact that training data and models were obtained only with sustained vowel audio segments. In this way, any subject would need to produce such kind of phonation (held at least 5 s) to get a prediction of their heart rate, and that could be not always appropriate, especially for heart rate continuous monitoring.

Chapter 6

Conclusions

In line with the initial hypothesis, the results show that heart rate can be estimated from voice with good accuracy. This was reached by well-established large space acoustic feature extraction and support vector machine. In particular, regression analysis allowed to estimate heart rate with moderate errors. Besides the discussed limitations, implications of this study are rich. In fact, even if the performance of the described audio-based predictors are not comparable with the modern measurement tools used in clinic, the possibility to extract heart rate as a non-contact measure, potentially accessible by any smartphone or device with a microphone, would make this important information usable by a large community, especially with the prospect of audio-based heart rate extraction from speech (e.g. from a conversation). Indeed, future works should aim to define new methodologies to seek for the complex relationship that links subtle variation on voice acoustic features induced by heart activity. Substantiating this relationship would represent a huge improvement in the field of emotion recognition, where standard acoustic features would be sided by physiological "high level features". Stress, fear, excitement and many other emotional states would be automatically recognized with way more reliability, since the effects of emotional states on voice can be controlled or hidden to a certain extent, while physiological effects of heart pumping would affect voice (in normal conditions) always in the same way. Not only emotion recognition, but also other fields, like telemedicine would profit from such technology. However, the downside is often to find the exact acoustic feature that would describe the variation of such

voice biomarker. Therefore, the future studies should converge on the design and implementation of a set of specified features best representing voice biomarkers, as well as the application of more refined digital signal processing and deep learning algorithms to improve performances and allow automatic recognition close to real-time. In fact, rather than estimate heart rate on defined audio chunks (e.g. sustained vowels), the reconstruction of a simplified version of the ECG signal (i.e. QRS complexes only) from voice, made at sample or frame level, would allow online heart rate extraction, without the need of producing the audio chunk first. In this direction, considering that the present dataset contains both audio and ECG time-series, prospective improvements of the present study would come from the application of more refined network architectures, like LSTM neural networks. Moreover, more and more data should be collected from a broader population of subjects following the proposed experimental protocol, in order to obtain abundance of training data as well as to capture all the nuances of the complex relationship between voice and heart rate. Altogether, the study finds that heart rate audio-based recognition has the potential to be widely used by a large public, aligning to the trend of today's healthcare: digital, unintrusive, immediate and for everyday monitoring.

Bibliography

- [1] J. S. Armstrong, *Long-range Forecasting: From Crystal Ball to Computer*, 2nd. ed. Wiley, 1985.
- [2] H. Anıışhan, *Estimation of heartbeat rate from speech recording with hybrid feature vector (HVF)*, Biomedical Signal Processing and Control, 49, 483-492, 2019.
- [3] J. M. Bland, D. G. Altman. *Statistical methods for assessing agreement between two methods of clinical measurement*, International Journal of Nursing Studies, 47, 931-936, 2010.
- [4] G. Berntson, J. Bigger, D. Eckbarg et al., *Heart Rate Variability: Origins, Methods, and Interpretative Caveats*, Psychophysiology, 34(6): 623-648, 1997.
- [5] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, Belmont, Massachusetts, 1995.
- [6] F. P. Branca, *Fondamenti di Ingegneria Clinica*, Springer-Verlag, Vol. I, 2000.
- [7] H. C. Burger, J. B. Van Milaan, *Heart-vector and Leads*, Heart, 8(3), pp. 157-161, 1946.
- [8] C. J. Burges, *A tutorial on support vector machines for pattern recognition*, Data Mining and Knowledge Discovery, vol. 2, no. 2, pp. 121-167, 1998.
- [9] J. Brank, M. Grobelnik, N. Milić-Frayling, and D. Mladenić, *Feature Selection Using Support Vector Machines*, Proc. of the 3rd Int. Conf. on Data Mining Methods and Databases for Engineering, Finance, and Other Fields, Bologna, Italy, 2002.

- [10] P. M. Chauhan, N.P. Desai, *Mel Frequency Cepstral Coefficients (MFCC) Based Speaker Identification in Noisy Environment Using Wiener Filter*, Green Computing Communication and Electrical Engineering (ICGCCEE), 2014 International Conference on. IEEE, pp. 1-5, 2014.
- [11] Y.-W. Chang, C.-J. Lin, *Feature ranking using linear SVM*, Journal of Machine Learning Research, Proceedings Track. 3. 53-64, 2008.
- [12] L. Cohen, *Time-Frequency Analysis: Theory and Applications*, Prentice Hall Signal Processing Series, Prentice Hall, 1995.
- [13] C. J. De Luca, *The Use of Surface Electromyography in Biomechanics*, Journal of Applied Biomechanics, 13(2): 135-163, 1997.
- [14] F. Eyben, *Real-time Speech and Music Classification by Large Audio Feature Space Extraction*, Springer Theses, 2016.
- [15] F. Eyben, M. Wöllmer, B. Schuller, *openSMILE - The Munich Versatile and Fast Open-Source Audio Feature Extractor*, Proc. ACM Multimedia, Florence, Italy, pp. 1459-1462, 2010.
- [16] F. Eyben, F. Wening, F. Gross, B. Schuller, *Recent Developments in openSMILE, the Munich Open-Source Multimedia Feature Extractor*, Proc. ACM Multimedia (MM), Barcelona, Spain, October 2013.
- [17] A. Evans, F. M. C. Boonstra, T. Perera, T., A. P. Vogel, S. C. Kolbe, H. Butzkueven, A. van der Walt (2018), *What Speech Can Tell Us: a Systematic Review of Dysarthria Characteristics in Multiple Sclerosis*, Autoimmunity Reviews, 17(12), 1202-1209, 2018.
- [18] R. Fletcher, *Practical Methods of Optimization*, Wiley-Interscience New York, II edition, 1987.
- [19] E. Frank, *General Theory of Heart-Vector Projection*. Circulation Research, 2(3), 258-270, 1954.

- [20] I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, Vol. 1. MIT press Cambridge, 2016.
- [21] D. Gabor, *Theory of communication*, Journal of the Institution of Electrical Engineers - Part I: General, Vol. 94, 1947.
- [22] J. Gao, B. M. Gurbaxani, J. Hu, K. J. Heilman, V. Emanuele II, G. F. Lewis, M. Davila, E. R. Unger, J. S. Lin, *Multiscale analysis of heart rate variability in non-stationary environments*, Frontiers in Physiology, vol. 4, 2013.
- [23] S. Haykin, *Neural Networks and Learning Machines*, Pearson, Third Edition, November 2008.
- [24] J. Hart, *Normal resting pulse rate ranges*, Journal of Nursing Education and Practice, 5(8), 2015.
- [25] T. Hastie, R. Tibshirani, e J. Friedman, *The Elements of Statistical Learning*, Springer New York, 2009.
- [26] G. James, D. Witten, T. Hastie, e R. Tibshirani, *An Introduction to Statistical Learning*, Springer, New York, 2013.
- [27] R. E. Klabunde, *Cardiovascular Physiology Concepts*, Lippincott Williams & Wilkins, Second Edition, 2012.
- [28] B.-U. Köhler, C. Hennig, e R. Orglmeister, *The principles of software QRS detection*, IEEE Engineering in Medicine and Biology Magazine, vol. 21, n. 1, pp. 42-57, 2002.
- [29] H. W. Kuhn, A. W. Tucker, *Nonlinear programming*, Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, Berkeley, CA, pp. 481-492, 1951.
- [30] X. Lu, M. Pan, Y. Yu, *QRS Detection Based on Improved Adaptive Threshold*, Journal of Healthcare Engineering, ID 5694595, 2018.
- [31] Y. Liu, *Hilbert Transform and Application*, Fourier Transform Applications, S. Salih, pp. 291-299, 2012.

- [32] S. Mahdiani, V. Jeyhani, M. Peltokangas, A. Vehkaoja, *Is 50 Hz High Enough ECG Sampling Frequency for Accurate HRV Analysis?*, Conference Proceedings IEEE Engineering in Medicine and Biology Society, 5948-51, 2015.
- [33] F. Marvasti, *Nonuniform Sampling, Theory and Practice*, Kluwer Academic/Plenum Publishers, New York, 2000.
- [34] J. Mercer, *Functions of Positive and Negative Type, and their Connection with the Theory of Integral Equations*, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol. 209, n. 441-458, pp. 415-446, 1909.
- [35] A. Mesleh, D. Skopin, S. Baglikov, A. Quteishat, *Heart Rate Extraction from Vowel Speech Signals*, Journal of Computer Science and Technology, 27(6), 1243-1251, 2012.
- [36] F. Q. Nuttall, *Body Mass Index*, Nutrition Today, 50(3), 117-128, 2015.
- [37] R. F. Orlikoff, R. J. Baken, *The Effect of the Heartbeat on Vocal Fundamental Frequency Perturbation*, Journal of Speech and Hearing, vol. 32, no. 3, pp. 576-582, 1989.
- [38] J. Pan, W. J. Tompkins, *A Real-Time QRS Detection Algorithm*, IEEE Transaction on Biomedical Engineering, vol. BME-32, no. 3, 1985.
- [39] M. Z. Poh, D. J. McDuff, R. W. Picard, *Advancements in noncontact, multiparameter physiological measurements using a webcam*, IEEE Transaction on Biomedical Engineering, vol. 58, no. 1, pp. 7-11, 2012.
- [40] M. Raj, *UART Receiver Synchronization: Investigating the Maximum Tolerable Clock Frequency Deviation*, Indian Journal of Science and Technology, Vol. 10(25), 2017.
- [41] B. Rammstedt, O. P. John, *Measuring personality in one minute or less: A 10-item short version of the Big Five Inventory in English and German*, Journal of Research in Personality, Vol. 41, pp. 203-212, 2007.
- [42] S. Raschka, *Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning*, arXiv:1811.12808 [cs.LG], 2018.

- [43] A. Ryskaliyev, S. Askaruly, A. P. James, *Speech Signal Analysis for the Estimation of Heart Rates Under Different Emotional States*, Proceedings of International Conference on Advances in Computing, Communication and Information, 2016.
- [44] M. Sakai, *Estimation of Heart Rate from Vocal Frequency Based on Support Vector Machine*, International Journal of Advances in Scientific Research, 2(01): 016-022, 2016.
- [45] S. Sammito, I. Böckelmann, *Factors Influencing Heart Rate Variability*, International Cardiovascular Forum Journal, pp. 18-22, 6, 2016.
- [46] E. Sejdić, I. Djurović, J. Jiang, *Time-frequency feature representation using energy concentration: An overview of recent advances*, Digital Signal Processing, 19(1), pp. 153-183, 2009.
- [47] B. Schölkopf, A. J. Smola, *Learning with Kernels*, MIT Press, 2002
- [48] B. Schuller, F. Friedmann, F. Eyben, *Automatic Recognition of Physiological Parameters in the Human voice: Heart Rate and Skin Conductance*, Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on. IEEE, pp. 7219-7223, 2013.
- [49] C. G. Scully, J. Lee, J. Meyer, A. M. Gorbach, D. Grandquist-Fraser, Y. Mendelson, K.H. Chone, *Physiological parameter monitoring from optical recordings with a mobile phone*, IEEE Transaction of Biomedical Engineering, vol. 59, no. 2, pp. 303-306, 2012.
- [50] S. Shalev-Shwartz, S. Ben-David, *Understanding Machine Learning*, Cambridge University Press, 2009.
- [51] D. U. Silverthorn, *Human Physiology, An Integrated Approach*, Pearson/Benjamin Cummings, 6th Edition, San Francisco, 2007.
- [52] S. Skodda, W. Grönheit, C. Lukas, B. Bellenberg, S. M. Von Hein, R. Hoffmann, C. Saft, *Two different phenomena in basic motor speech performance in premanifest Huntington disease*, Neurology, 86(14), 1329-1335, 2016.

- [53] J. Smith, A. Tsiartas, E. Shriberg, A. Kathol, A. Willoughby, M. de Zambotti, *Analysis and Prediction of Heart Rate Using Speech Features from Natural Speech*, ICASSP, pp. 989-993, 2017.
- [54] R. P. Smith, J. Argod, J. Pépin, P. A. Lévy, *Pulse transit time: an appraisal of potential clinical applications*, Thorax, 54(5): 452-457, 1999.
- [55] A. J. Smola, B. Schölkopf, *A Tutorial on Support Vector Regression*, Statistic and Computing 14: 199-222, 2004.
- [56] S. Solnik, P. Rider, K. Steinweg, P. DeVita, T. Hortobágyi, *Teager-Kaiser energy operator signal conditioning improves EMG onset detection*, European Journal of Applied Physiology, 110: 489-498, 2010.
- [57] V. Vapnik, C. Cortes, *Support Vector Networks*, Machine Learning, 20, 273-297, 1995.
- [58] V. Vapnik, *Statistical Learning Theory*, Wiley-Interscience, 1998.
- [59] J. Wagner, T. Baur, Y. Zhang, M. F. Valstar, B. Schuller, E. André, *Applying Cooperative Machine Learning to Speed Up the Annotation of Social Signals in Large Multimodal Corpora*, arXiv:1802.02565 [cs.HC], 2018.
- [60] I. H. Witten, E. Frank, *Data Mining*, Elsevier, Morgan Kaufmann Publishers, San Francisco, 2005.

ACKNOWLEDGEMENTS

First, I would like to thank my thesis adviser Prof. Stefano Squartini for the huge opportunity of developing this thesis in the field of computational audio processing with the ERASMUS program. He was an outstanding adviser, providing me with countless notions and insights into his field of expertise, as well as leaving me the freedom of choosing a thesis project strongly connected with biomedical engineering. Also, his mediation with contacts of the receiving institution was fundamental for planning successfully my traineeship abroad, as well as his continuous supervision, helping me considerably with the writing progression of the thesis. I would like also to thank with all my heart audeERING's key persons: Dagmar Schuller and Prof. Björn Schuller, who – as respectively CEO and CSO of the company – made my stay possible in first place, and my coadviser Florian Eyben, who – as audeERING's CTO – followed always my work with true interest, providing me with consistent insights and contributions on the development, implementation and research of the thesis project. It is my wish to thank also audeERING's former colleagues for their help, support and suggestions throughout my traineeship, in particular: Simone Hantke, Christoph Hausner, Taewoo Kim, Christopher Oates, Soroosh Mashal, and Hesam Sagha. Special thanks go also to Stefano Cardarelli, Alessandro Mengarelli and Andrea Tigrini, my university colleagues from the information engineering department, Laboratorio di Analisi del Movimento, for their availability for insightful exchange and inspiring discussions. Most of all, my utmost gratitude goes to my family, to my parents and my girlfriend, for their undying love and support, and their doubtless trust they always put in me and my choices.

