SNR-Dependent Models for Parkinson's Disease Detection

Master's Thesis

Sergi Escanes

Aalborg University
Sound and Music Computing

Sound and Music Computing
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
SNR-dependent models for Parkinson's disease detection

Theme:
Sound and Music Computing

Project Period:
Spring Semester 2019

Project Group:
18831

Participant(s):
Sergi

Supervisor(s):
Mads Græsbøll Christensen
Amir Hossein Poorjam Alavijeh

Copies: 1

Page Numbers: 56

Date of Completion:
May 27, 2019

Abstract:

Parkinson's disease (PD) is one of the most common neurodegenerative disorders. It can be diagnosed by analyzing short fragments of speech recorded from potential patients. This master's thesis aims at studying the impact of noise when diagnosing PD through recorded speech. The mismatch between training and test condition is one of the main sources of detection error in speech-based applications. In this project, the following hypothesis is tested: matching between training and test conditions at model level can improve the PD detection accuracy. To this aim, the impact of training and test SNR mismatch on the PD recognition performance is studied and a SNR-dependent model is trained.

Sound and Music Computing
Aalborg Universitet
http://www.aau.dk

AALBORG UNIVERSITET
STUDENTERRAPPORT

Titel:
SNR-afhængige modeller til Parkinsons sygdomsdetektering

Tema:
Lyd og Musik Computing

Projektperiode:
Forårssemester 2019

Projektgruppe:
18831

Deltager(e):
Sergi Escanes

Vejleder(e):
Mads Græsbøll Christensen
Amir Hossein Poorjam Alavijeh

Oplagstal: 1

Sidetal: 56

Afleveringsdato:
27. maj 2019

Abstract:

Parkinsons sygdom (PD) er en af de mest almindelige neurodegenerative sygdomme. Det kan diagnosticeres ved at analysere korte fragmenter af tale optaget fra potentielle patienter. Master afhandlingen sigter på at studere virkningen af støj ved diagnosticering af PD gennem indspillet tale. Misforholdet mellem træning og testtilstand er en af de vigtigste kilder til detekteringsfejl i talebaserede applikationer. I dette projekt, den følgende hypotese testes: Sammenligning mellem træning og testforhold på modelniveau kan forbedre PD-detektering nøjagtighed. Til dette formål studeres virkningen af træning-test SNR-mismatch på PD-anerkendelsespræstationen, og en SNR-afhængig model trænes.

Contents

This master's thesis is dedicated to my grandfather, who passed recently victim of Parkinson's disease.

Acknowledgments

I would like to thank Mads Græsbøll Christensen and Amir Hossein Poorjam Alavijeh for their input in this master's thesis.

Chapter 1

Introduction

With the progress of medicine, more and more people are nowadays capable of living really long lives. Life expectancy has reached above 80 years in some parts of the world [30]. These are indeed good news for humankind, but not everything is as good as it seems at first glance. Longer lives do not necessarily mean better ones.

It is precisely because life expectancy is growing that diseases that tend to affect the elder are becoming more important to be studied and understood. Although the heart disease remains the main responsible for human deaths (in 2017, [34]), the victims of cancer grow larger and larger by the day. While the first could be reduced greatly by adopting healthier habits, the second remains a challenge due to its genetic component and yet to find cure.

But this thesis tackles neither of those diseases. The main focus of this work is on a specific kind of neurodegenerative disease: Parkinson's disease (henceforth PD). Early symptoms of the disease may include tremor in hands or legs and feet and a general slowness of the body while moving or doing basic activities. The disease is degenerative, meaning that the symptoms grow worse and worse with time. Late symptoms include generally rigidity and loss of other basic functions of the body such as walking or speaking [31]. This severely affects the patients, who live very restricted lives and often need another person to look after them most of the time as well as psychological help. Patients often require regular visits by their physician every 4-6 months [11]. Many treatments have been developed and are currently used to alleviate the aforementioned symptoms. Unfortunately, to date, there is no cure for the disease. Not only that, but often these treatments carry side effects that may hinder even more the lives of the patients.

PD is responsible of making the lives of its patients miserable, not only for them, but also for their families. Since it mainly involves the brain, it however remains very challenging. That is why it is of the utmost importance to dedicate efforts and resources to fight this cruel disease. This thesis builds upon this idea and collaborates in that aspect.

1.1 Aim of the Research

This thesis uses a tool found in a project that was launched by PatientsLikeMe, a health network with more than 650,000 members [26] and Sage Bionetworks [24], a non-profit research organization.

These organizations launched a mobile platform aimed at gathering information from patients to better track their symptoms and change their treatment accordingly . The platform is called mPower [11] and it used Apple's ResearchKit library to evaluate the current status of the patients via something as simple as an iPhone. The platform includes many surveys and activities including memory, walking and voice as well as basic information (demographics) and PD-related questions created by expert physicians. In figure 1.1, a table including different questions and possible answers is presented:

Question	Variable name	Variable details
Due to Parkinson's disease, how often during the last month have you had difficulty getting around in public?	PDQ8-1	one of: {'Never', 'Occasionally', 'Sometimes', 'Often', 'Always'}
Due to Parkinson's disease, how often during the last month have you had difficulty dressing yourself?	PDQ8-2	one of: {'Never', 'Occasionally', 'Sometimes', 'Often', 'Always'}
Due to Parkinson's disease, how often during the last month have you felt depressed?	PDQ8-3	one of: {'Never', 'Occasionally', 'Sometimes', 'Often', 'Always'}
Due to Parkinson's disease, how often during the last month have you had problems with your close personal relationships?	PDQ8-4	one of: {'Never', 'Occasionally', 'Sometimes', 'Often', 'Always'}
Due to Parkinson's disease, how often during the last month have you had problems with your concentration, e.g. when reading or watching TV?	PDQ8-5	one of: {'Never', 'Occasionally', 'Sometimes', 'Often', 'Always'}
Due to Parkinson's disease, how often during the last month have you felt unable to communicate with people properly?	PDQ8-6	one of: {'Never', 'Occasionally', 'Sometimes', 'Often', 'Always'}
Due to Parkinson's disease, how often during the last month have you had painful muscle cramps or spasms?	PDQ8-7	one of: {'Never', 'Occasionally', 'Sometimes', 'Often', 'Always'}
Due to Parkinson's disease, how often during the last month have you felt embarrassed in public due to having Parkinson's disease?	PDQ8-8	one of: {'Never', 'Occasionally', 'Sometimes', 'Often', 'Always'}

Figure 1.1: PD assessment questionnaire taken from [11] and based upon [5].

From this observational study, it is of our interest to focus on the part related to the voice.

In the voice activity patients were asked sustained vowels at a comfortable pitch and intensity during 10 seconds. The corresponding signals were then recorded.

Taking the data gathered by this part of the study, one can build models that include the characteristics of the speech present in PD patients. Afterwards, a PD detection system can be created to evaluate new potential patients. This is key due to the fact that an early diagnosis is crucial for the patients to start a treatment according to their symptoms. This all sounds promising, but there is a main challenge that has to be dealt with in order to achieve good enough results for these models to be used in real life. This challenge is the difference between the speech characteristics of the signals used for building the models and the signals used for actually testing the model. When these conditions are mismatched, usually performance drops [13]. These conditions include different acoustic environments, communication channels and types and levels of noise and distortion. In other words, conditions should be matched for maximizing performance.

That would be the first step towards a remote PD detection system. A system as such would be useful to monitor the patients' symptoms and avoid the need of visiting a doctor by means of a non-invasive method. Moreover, as discussed in [11], recording their voices periodically is a way (among others) of keep the patients engaged in an activity aimed at alleviating their symptoms. The main drawback of a remote control system is that the conditions under which the signals are recorded are not controlled, and it is thus difficult to make them match with the ones given in the training data. Hence, studies aimed at solving this kind of problem, such as this thesis, are relevant and worth investing time in.

1.2 Motivation

The motivation behind this thesis is thus investigating on mismatched conditions in PD detection, making them more robust against changes in the operating environments. More specifically, the fact that motivates this work is providing with experimental proof that when under matched conditions of training and testing, the performance of the algorithms aimed at detecting the disease increases compared to mismatched conditions. This is all done to transform the model to be matched with the acoustic characteristics of the test signal.

1.3 Description of the problem

The problem dealt with in this thesis is PD detection from voice. That is, given a voice signal, classify it into healthy or PD. A problem of binary classification is often called detection.

Starting from raw signals which contain uttered vowels from a speaker, relevant features that capt the information present on speech are extracted. Later on, these features are fed to a classifier or detector that emits a verdict of whether that person has or not the disease, with a certain probability of success.

According to [18], noise is the main factor that makes the conditions between the training and test signals mismatch. One approach to solving this issue consists of training several models at different signal-to-noise ratios (henceforth SNRs), thus creating a SNR-dependent model or multi-SNR model.

Chapter 2

PD Detection Techniques

2.1 State-of-the-art

In this section some of the methods used for PD detection are hereby presented. A good literature review with regards to PD detection through voice can be found in [23]. In this study, they focused on articulatory and phonatory approaches to PD detection. The first makes use of any kind of sound coming from articulating with the mouth and insides while the second focuses on the resonant sound emitted using the vocal tract. For the first category, regular speech is employed, while for the second phonated speech is used.

In [35], researchers detected PD employing running speech using MFCCs in three different languages obtaining from 0.85 to 0.99 accuracy depending on the language and speech task. However, results went down to 0.6 when cross-validated, showing a tendency to overfit.

In [4], a study shows a UBM-GMM method for PD detection achieving 0.8 accuracy in non-noisy conditions and high-quality microphones. It also used MFCCs but was not solely based on phonated speech, for it included articulatory speech for better results. Another example of a GMM-UBM approach to PD detection can be found in [15].

A different approach is used in [7]. This is the i-vector method, described in section 2.1.2. Both GMM-UBM and i-vector approaches are used in [23]. In this study, they reached

0.86 ± 0.07 and 0.87 ± 0.09 accuracy for the GMM-UBM and the i-vector approaches, respectively and for phonated speech. A block diagram depicting the work flow in a PD detection system is presented in figure 2.1:

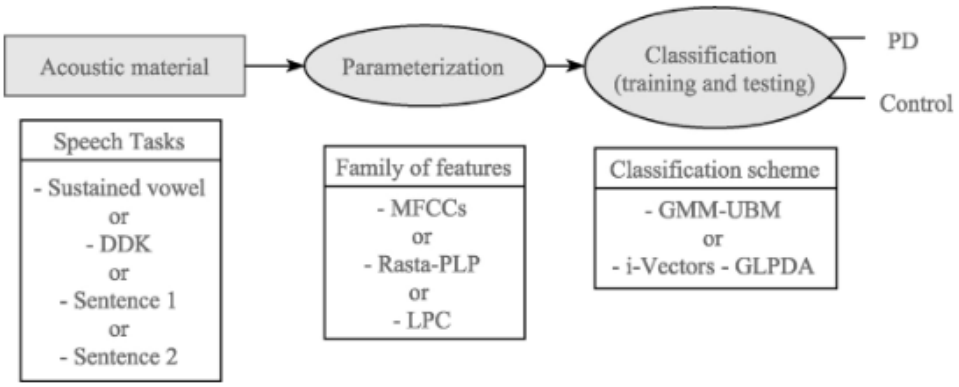

Figure 2.1: Block diagram scheming the methodology followed in a PD detection system. Taken from [23].

Note that in all the studies aforementioned, the presence of noise was almost non-existent, meaning high SNRs (not specified). Therefore, and to the best knowledge of the writer a multi-SNR approach to PD detection would mean a novel one on the subject.

2.1.1 Gaussian Mixture Models

Gaussian Mixture Models (henceforth GMM) are models based on a linear combination or mixture of Gaussian distributions. Each component of the mixture is represented by its vector mean and its covariance matrix. The philosophy behind the multivariate model is to model subpopulations within a given population.

The assumption then of normal distribution, which is quite common in many algorithms, makes this model apt to be used in a wide variety of applications such as unsupervised learning or clustering. It can be used in financial models [28] or for handwriting recognition [25]. Amongst its application, one can also find, of course, speaker recognition.

The main strength of this method is that relies on a well-known statistical model. Since it makes assumptions on data, any change made to the model is predictable. These assump-

tions can be also the weakness of this method, because they may not be entirely true or even far from it.

As a statistical model, GMMs make use of optimization algorithms to optimize their parameters. These are iterative methods that keep updating the parameters in the model to maximize (or minimize) certain variables. The algorithm used when working with GMMs is the Expectation-Maximization (henceforth EM) [8] algorithm. More details on this algorithm and GMMs are given in section 2.2.

2.1.2 Joint Factor Analysis and i-Vector

Another approach to channel compensation is joint factor analysis (henceforth JFA). This model considers speech as a series of components added up together called supervector: a speaker-dependent, a speaker-independent, a channel dependent, and a residual component. This way, the undesired variability of the model is explicit and thus removing its undesired parts becomes easier. That is why this more recent approach has become state-of-the-art in speaker recognition [12]. This method applied to pathological voices is presented in recent work in AAU's Audio Analysis led by Prof. Mads G. Christensen in collaboration with Dr. Max A. Little in [1].

A more recent and simpler approach is the i-vector method [7]. This approach keeps the speaker-independent component of the JFA approach and models the rest as a product of a variability component times the i-vector. The principle behind the i-vector is the same as in JFA, but in the i-vector approach both the speaker-dependent and the channel components in a single one that accounts for the total variability. This new approach was motivated by experimental results from [6]. In this work, it was found out that the channel factors modeled in the JFA also contain information of the speaker.

2.2 Model Used: GMM-UBM Model

The model or the algorithm used for PD detection throughout this thesis is a GMM-UBM model. Overall, the main reason why this was the model chosen for this thesis is its extended use in other works such as [23] for the specific problem of PD detection. This section describes how it works.

The GMM-UBM-based approach can be divided in several stages. Fist, a Universal Background Model or UBM is created. This is done training a GMM-based model on data that contains speech from both genders at different ages. It is important that the database used for creating the UBM is well balanced to avoid biasing the system. This point will be discussed more deeply in section 4.1. The aim of the UBM is to create a general model for all the speakers, that will be later on adapted to the new speakers presented to the algorithm. This adaptation part is key, and is the second stage of the process. During this second phase, a hypothesized speaker model is created. The third and last phase involves comparing the two models: the universal and the hypothesized. This is done computing the log-likelihood ratio, which gives an score that represents how similar the two models are. If the likelihood of an observation given the PD model is close to that of the background model, the likelihood ratio of the observation is small and falls below the threshold, then it is rejected. The same applies to the model for the healthy control group.

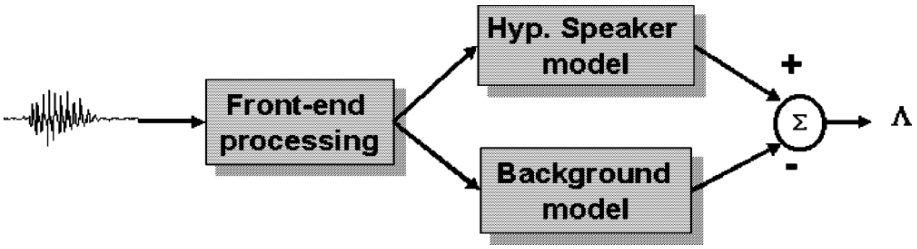

Figure 2.2: Block diagram of the GMM-UBM-based approach used. Taken from [15].

This is the overview of this chapter. It can be seen summarized in fig 2.2. Let us see all three stages one by one in the following sections. This section follows [15].

2.2.1 Universal Background Model

As mentioned, the UBM is built upon a GMM using the EM algorithm. Let us see how it works.

A GMM is described by a series of mixtures or weighted sum of M components and its mixture density is

$$p(x|\lambda) = \sum_{i=1}^{M} w_i p_i(x) \tag{2.1}$$

where $p_i(x)$ is the ith unimodal Gaussian density characterized by its vector mean μ_i of size $D \times 1$ and its covariance matrix $\sum_i$:

$$p_i(x) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} \exp[-\frac{1}{2}(x-\mu_i)^T(\Sigma_i)^{-1}(x-\mu_i)] \tag{2.2}$$

where x is a D-dimensional feature vector and w_i are the weights of the mixture, which fulfills

$$\sum_{i}^{M} w_i = 1 \tag{2.3}$$

The parameters that represent the model are thus denoted as $\lambda = \{w_i, \mu_i, \Sigma_i\}$.

The series of parameters presented are estimated using the maximum likelihood model. That is done employing the EM algorithm [8]. The work-flow of this algorithm consists of defining the likelihood that works as a score, which estimates how accurate the model describes the data. After each iteration of the algorithm, and with the aim of increasing the likelihood, the parameters of the GMM are thus refined and refined until a sufficient number of iterations are done. Typically, five iterations are used. This whole process is represented in the following equation:

$$p(x|\lambda^{k+1}) > p(x|\lambda^k) \tag{2.4}$$

where k is the iteration number. For a more in-depth description of the algorithm, please see [36].

2.2.2 Adaptation

In the second stage of the process, the hypothesized speaker model is derived adapting the UBM model to a new one. This process thus updates the parameters estimated before in the GMM using data from a new speaker. This idea is depicted in figure 2.3 and it is know as Maximum a Posteriori or MAP estimation.

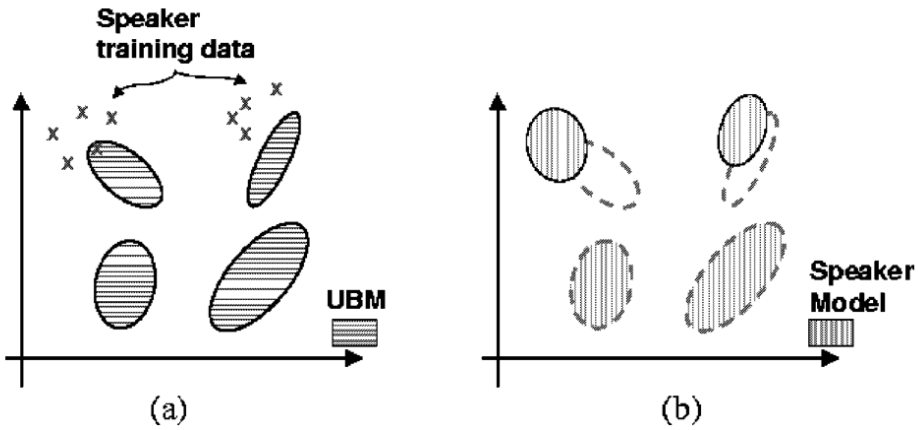

Figure 2.3: Parameters of the universal model depicted on a) and adapted mixture parameters for the speaker model depicted in b). Taken from [15].

That is done using the following equations. Given series of training vectors $X = (x_1, ..., x_T)$ from a hypothesized speaker and a UBM model the following quantity is computed:

$$Pr(i|x_t) = \frac{w_i p_i(x_t)}{\sum_j w_j p_j(x_t)} \tag{2.5}$$

where i refers to the ith mixture and x_t to the tth training vector. The quantity $Pr(i|x_t)$ expresses how well the training data aligns to the UBM model in terms of probability. After this, the new weight $\hat{w}_i$, mean $\hat{\mu}_i$, and variance $\hat{\sigma}_i$ vectors for the ith mixture component are given by

$$\hat{w}_i = [\alpha_i^w n_i/T + (1-\alpha_i^w)w_i]\gamma \tag{2.6}$$

$$\hat{\mu}_i = \alpha_i^m E_i(x_t) + (1-\alpha_i^m)\mu_i \tag{2.7}$$

$$\hat{\sigma}_i^2 = \alpha_i^v E_i(x_t^2) + (1-\alpha_i^v)(\sigma_i^2 + \mu_i^2) - \hat{\mu}_i^2 \tag{2.8}$$

where $\alpha_i^w, \alpha_i^m, \alpha_i^v$ are the adaptation coefficients which control the balance between the new and the old estimates and γ is a scaling factor that ensures that the sum adds to one. The rest of the parameters are:

$$n_i = \sum_t Pr(i|x_t) \tag{2.9}$$

$$E_i(x_t) = \frac{1}{n_i}\sum_t Pr(i|x_t)x_t \tag{2.10}$$

$$E_i(x_t^2) = \frac{1}{n_i}\sum_t Pr(i|x_t)x_t^2 \tag{2.11}$$

An extra parameter is defined:

$$\alpha_i^\rho = \frac{n_i}{n_i + r^\rho} \tag{2.12}$$

This allows for an adaptation depending on the mixture, for not all mixtures are adapted or updated in the same amount. Note that if n_i is low, then $\alpha_i^\rho \to 0$. That means that given a low probabilistic count of the ith mixture, there is an emphasis of the old and better trained (because of the presence of more data) parameters. Otherwise, those are deemphasized. Therefore, the relevance factor r^ρ is a good way how much data is needed to start replacing the old estimates by the new.

2.2.3 Log-likelihood Ratio

The main feature of the likelihood function within a GMM model is that it is based on a well known statistical model and is insensitive to variations in time for text-independent task, as it is discussed in [36].

After creating the UBM model and the adapted one, they are compared using what is called a likelihood test:

$$\frac{p(Y|H_0)}{p(Y|H_1)} \geq \Theta \quad \text{accept } H_0 \tag{2.13}$$

$$\frac{p(Y|H_0)}{p(Y|H_1)} < \Theta \quad \text{reject } H_0 \tag{2.14}$$

where $p(Y|H_i)$ with $i = 0,1$ is the probability density function (henceforth PDF) for the hypothesis test H_i and the speech signal Y, with the following hypothesis test:

$$H_0 : \text{Y belongs to the hypothesized speaker S} \tag{2.15}$$

$$H_1 : \text{Y does not belong to the hypothesized speaker S} \tag{2.16}$$

$\Theta$is the threshold in the likelihood ratio.

However, when working with Gaussian distributions, which belong to the group of the exponential functions, the log-likelihood ratio is used instead. Let λ_{hyp} and $\lambda_{h\hat{y}p}$ be the models that represent the test hypotheses H_0 and H_1, respectively. Hence, the log-likelihood ratio becomes

$$\Lambda(X) = \frac{\log p(X|\lambda_{hyp})}{\log p(X|\lambda_{h\hat{y}p})} = \log p(X|\lambda_{hyp}) - \log p(X|\lambda_{h\hat{y}p}) \tag{2.17}$$

with $X = (x_1, ..., x_T)$ being T different feature vectors and λ the model represented by the weights, means, and covariances of the mixture. Its PDF is computed as in 2.1. The log likelihood is thus

$$\log p(X|\lambda) = \frac{1}{T} \sum_{t=1}^{T} \log p(x_t|\lambda) \tag{2.18}$$

where the term $1/T$ is included for normalization as done in [21]. Furthermore, this term aims at compensating for the wrong assumption of independence among the feature vectors, which results in an underestimation of the actual likelihood.

In the end of this process a GMM-UBM model is created. This thesis focuses on a multi-SNR approach. That means that several models are created to model speakers at different SNRs. The SNRs chosen were -10, 0, 10, 20 and 30dB. Preliminary experiments showed that a jump of 5db SNR was too small of a jump to get significantly different results. Therefore, a total of 5 models were created representing noisy conditions (the first three models) and less noisy (or somewhat clean) conditions (the last two). Each model was trained on a different SNR and was evaluated at the same time on all of them giving a total of 5 different models tested on 5 different SNRs.

Chapter 3

Front-End Processing

As in many other scenarios, data is not presented raw (meaning unprocessed) to an algorithm. Prior to classification, data is undergoes a process during which raw data is transformed into what are called features. This process is also called front-end processing. The goals behind extracting features from data are mainly two. Firstly, to facilitate the job of the classifier (or any other kind of algorithm for that matter) by getting rid of redundant or unnecessary information. Secondly, to reduce sample size. The reason of latter can be obvious: the less data one has, the less time one needs to spend processing it for, for instance, training a classifier. The former may not seem that obvious. Let us delve in it a bit more and see how it applies to the problem at hand.

The speech uttered by people is recorded and stored in files. These files contain signals sampled at a certain sampling frequency and distributed in one or more channels. The signal thus consists of a certain number of samples per channel, and each contain an amplitude of the sound uttered at that very moment. This is a time representation of the signal, or what we call a raw signal. However, humans do not perceive sound in this kind of representation. As we know, the auditory system transforms the sound into a series of electric impulses that are later on interpreted by the brain as the sound that is emitted in the first place. This process undergoes within the ear, that vibrates depending on the frequency of the sounds uttered. Therefore, the ears are responsible for changing a time representation of the sound into what we call a frequency representation (see figure 3.1). The following techniques in charge of converting time representations into frequency representations are presented in the next sections.

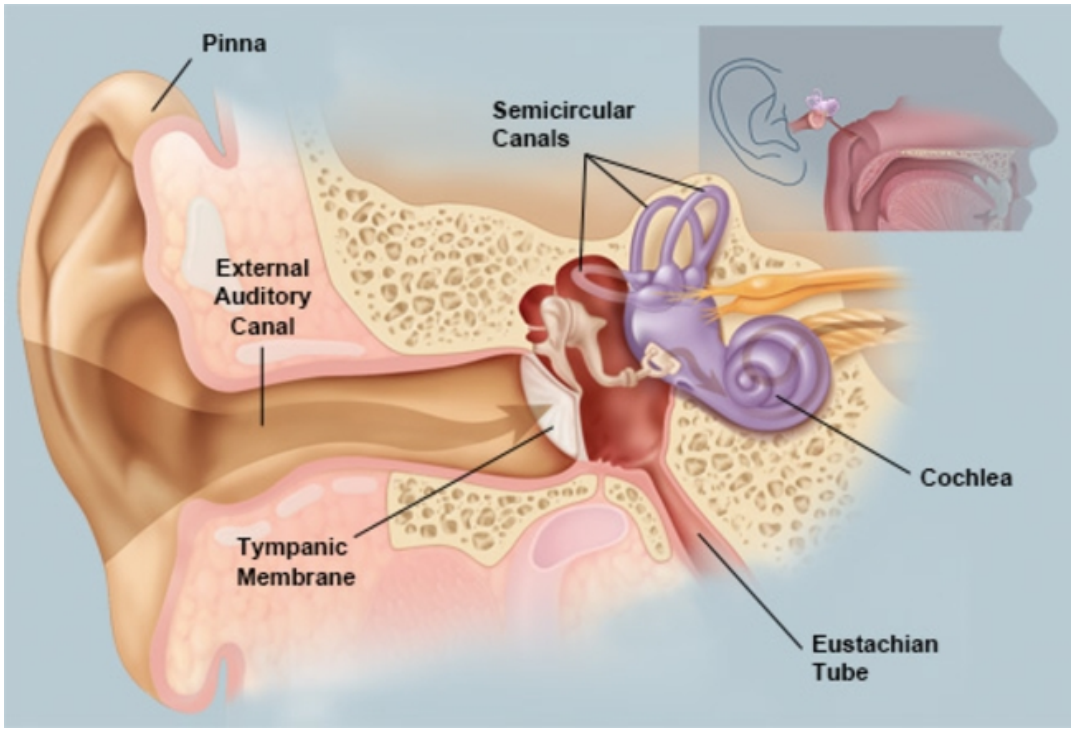

Figure 3.1: Picture of the human ear. The sound waves enter the ear canal and make the tympanic membrane vibrate. This vibration is transmitted to the cochlea and gets to the brain through the auditory nerve (right part of the picture). Taken from [37].

3.1 State-of-the-art

The most popular front-end processing techniques applied to speech analysis for PD detection that are used in the literature are the Linear Predictive Coding Features (henceforth LPCs), Perceptual Linear Predictive Features (PLPs) and the Mel Frequency Cepstral Coefficients (henceforth MFCCs). The main reason these are used is because they contain clinically meaningful features used in voice biometrics applications [36]. Let us see them in more detail. This section follows [36] and [27].

3.1.1 Linear Predictive Coding Features

Linear Predictive Coding Features or LPCs are mainly used for speech analysis and re-synthesis. They are often used for formant modeling. Formants are the part of the sound uttered that is due to the vocal tract and the mouth, that is, the cavities that resonate. Other kinds of sounds that can be uttered include sounds emitted by the tongue and the lips (sibilance). LPCs are used, for example, by phone companies for voice compression. It is also used for compression before transmission of wireless signals.

As mentioned, the vocal tract is modeled as a filter. It is based on the assumption that the vocal tract can represented using a series of nearly periodic pulses which are generated by the vocal cords. When unvoiced speech is present, the sounds emitted are generated via random noise, that emulates the turbulence of the air flowing out of the mouth through the vocal tract. Every sample is a linear combination of the last m samples that precede it. The factors present in the linear combination are the LPCs, and are computed minimizing the predictor error.

LPCs are quite a popular approach [33]. However, given their linear nature and knowing that the human speech is precisely non-linear, the do not seem to be the best approach to the problem at hand.

3.1.2 Perceptual Linear Predictive Features

Perceptual Linear Predictive Features or PLPs are the natural evolution of LPCs. These are also obtained using linear prediction, but incorporate notions from psychoacoustics to create a model that resembles more how humans perceive sound. PLPs discard information present on data that is not relevant to the human auditory transforming the characteristics of the sound to match those that are perceived by the human auditory system.

These features are explored in this thesis. A more in-depth description of them can be thus found in section 3.2.

3.1.3 Mel Frequency Cepstral Coefficients

Mel Frequency Cepstral Coefficients or (Log-Mel) Filterbank Energies are also features widely used in speech analysis. They also are state-of-the-art front-end processing technique in deep learning for sound detection and classification, as, for instance, in [10].

These features, are also derived borrowing notions from psychoacoustics. For instance, they incorporate the fact that the amplitude of sound is perceived not as linear but as logarithmic (it is called loudness). Or also the fact that humans cannot tell two too closed separated frequencies apart (that is why perceived frequency is often called pitch). The main advantage of these features is their ability to keep track of small changes in the signal due to their high sensitivity to noise and other variabilities, as explained in [2].

The main difference between PLPs and MFCCs are how the divide the frequency spectrum. This part is key because it accounts for most of the dimensionality reduction notion present in these techniques. Since humans can hear sound up to frequencies of 20,000 Hz best case scenario, modeling those would be expensive in computational terms. Taking the notion of pitch, however, the frequency spectrum is divided in different frequency bins, that represent the whole spectrum into different parts. How this division is made is where the difference between the methods rely. For PLPs, a linear scale is used for frequencies below 500 Hz and a logarithmic one above that frequency (often called Bark scale). As for MFCCs, a logarithmic scale (often referred to as mel scale) is used throughout. A more in-depth description of them can be found in [29]. These features seem to be worth investigating for PD detection. However, PLPs show better results [23] and are thus the ones to be used in this thesis.

3.2 Feature Extraction Method Used: PLPs

This section is destined to give more details on the theoretical aspects on the features used throughout the experiments carried out in this thesis. The main reason for using them is their high performance for the specific task of PD detection, as described in [23]. They are the Perceptual Linear Predictive features or PLPs were first presented on a paper from 1990 by Hynek Hermansy [32]. This section is based on this.

As mentioned in earlier sections (3.1.2), PLPs supposed an evolution to the Linear Predictive model. PLPs also use linear predictors, but add some changes to the power spectrum

of speech prior to that. These changes are based upon psychoacoustics, and are aimed at better modeling how humans perceive sound. The block diagram containing the processing of PLPs is shown in figure 3.2. Let us analyze and describe each block separately.

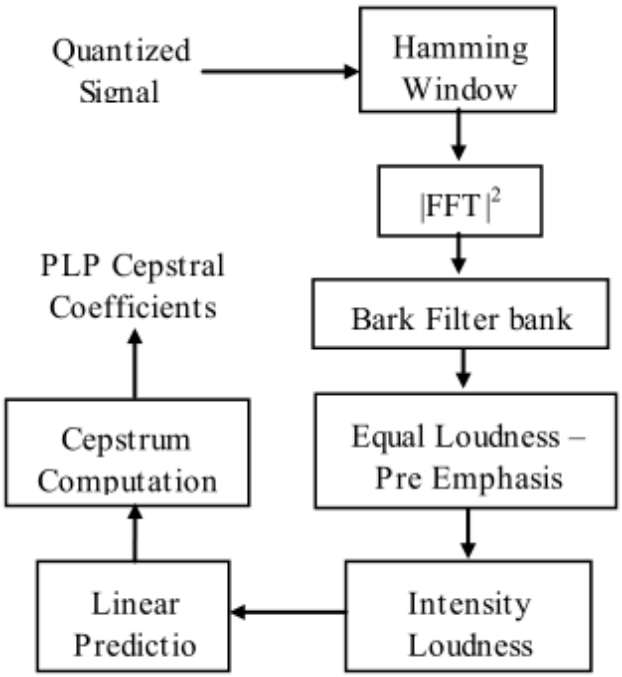

Figure 3.2: Block diagram of PLPs' process taken from [27].

3.2.1 Windowing

The signal is windowed using the Hamming window

$$w(n) = 0.54 + 0.46 \cos\left[2\pi n/(N-1)\right] \tag{3.1}$$

where N is the window's length.

The choice of the window is a typical one and widely used, for instance, when computing MFCCs. Rectangular windows have better frequency resolution, but at the cost of

spectral leakage. That is why this window is often chosen. Frequency resolution can be increased increasing the windows' size. Typical values for the windows' length are 20-30ms of speech. The step of windowing is necessary prior to computing the Fourier Transform (henceforth FT) with a finite number of samples.

3.2.2 Power Spectrum

Prior to computing the power spectrum, the Discrete Fourier Transform (henceforth DFT) is computed using the Fast Fourier Transform (henceforth FFT) algorithm. Typical values for the FFT are 256 points. Then, the power spectrum is computed:

$$P(w) = Re[S(w)]^2 + Im[S(w)]^2 \tag{3.2}$$

Where $Re[S(w)]$ and $Im[S(w)]$ are the real and imaginary parts of $S(w)$ respectively, the result of convolving the signal with the Hamming window.

3.2.3 Bark Filter Bank

After taking the power spectrum, it is convolved with what is known as the Bark filterbank. This filter organizes the spectrum in several bins taking into account how loudness is perceived by humans. It is similar to the filterbank applied when computing MFCCs, only the scale is different. According to their creators: "...a frequency scale on which equal distances correspond to perceptually equal distances" [40]. This translates to a filter that has a logarithmic shape above 500 Hz and is rather linear below that frequency. The shape of the filter is depicted in figure 3.3:

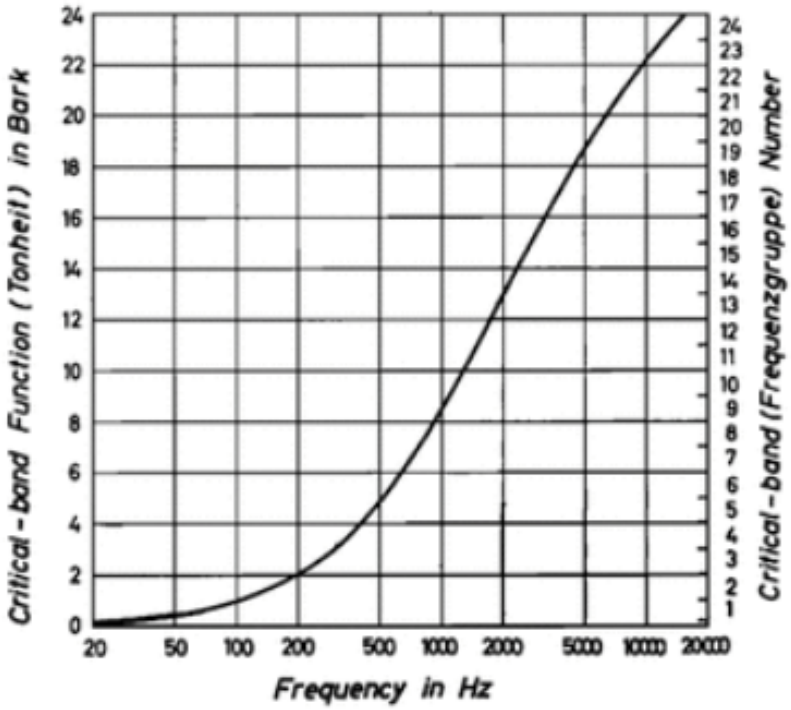

Figure 3.3: Bark filter [40]. Frequency in Hz on the x-axis and frequency bands in Bark on the y-axis

3.2.4 Equal Loudness Pre-Emphasis

To account for the fact that different frequencies are perceived differently. That is, to adapt the signal according to the sensitivity of the human hearing: high sensitivity in middle frequencies and poorer below 100Hz and above 10,000Hz approximately. For that, the signal is pre-emphasized at the same loudness (40 dB) with the following function [32]:

$$E(w) = \frac{(w^2 + 56.8 \times 10^6)w^4}{(w^2 + 0.38 \times 10^9)(w^6 + 9.58 \times 10^{26})} \tag{3.3}$$

3.2.5 Intensity Loudness

To account for the non-linear relation between the sound intensity and its perceived loudness, the values of the signal are raised to the power of 0.33, following [19].

3.2.6 Linear Prediction

As well as done in Linear Predictive Coding, linear prediction is applied also when computing PLPs. As described in section 3.1.1, every new factor is computed taking the last m samples that precede it. It is called linear prediction because the new samples are modeled as a linear combination. The parameters of this combination are set by minimizing the predictor error. For a more detailed description on the subject, please see [36].

3.2.7 Cepstral Computation

The elements of the resultant cepstrum are then computed by taking the Inverse Discrete Fourier Transform (henceforth IDFT).

A picture of the resultant PLP coefficient matrix is depicted in figure 4.3.

Figure 3.4: PLP spectrogram of a healthy female speaker at 30dB SNR. Time frames in the X-axis and linear PLP coefficients on the Y-axis.

3.2.8 Delta Cepstrum

So far, the procedure followed presents a way to give a representation of the spectral properties of the signal at a given time, that is why they are often called static features. However, it is interesting to see how these evolve. That is way dynamic features are also computed. These features are obtained taking the derivatives of the first, that is why they are often called delta features or delta cepstrum. Often, the delta features of the delta features are also computed. These involve the second derivatives of the static features.

This method applies to any kind of features: PLPs, MFCCs, etc. Throughout this thesis, both the delta and the delta-delta coefficients are used. For a more in-depth analysis on the topic, the reader is suggested to follow [36].

Chapter 4

Experimental Setup

4.1 Database

4.1.1 PD Database

The database containing the signals used throughout is the one referenced in Jenkinson et al.'s work of the mPower study, article published in the Nature magazine [11]. From that database and for the purposes of getting a balanced dataset, a subset is chosen. That subset contains data from a total of 800 people divided as follows:

- 200 records of healthy males
- 200 records of healthy females
- 200 records of PD diagnosed males
- 200 records of PD diagnosed females

The signals' duration are roughly 10 seconds, sampled at 44.1 kHz. They are mono signals. As mentioned in earlier chapters, for the study of pathological voices, speech in the form of phonation, that is, sustained vowels, is used instead of regular speech. This is due to two reasons. Firstly, sustained vowels reveal pathologies common in dysphonic speakers,

since they are not able to generate sustained speech [9]. Secondly, when patients are forced to utter sustained speech, "the confounding complexities of articulatory movement during speech are largely avoided" [38].

As intuition tells, the more balanced a database is with respect to the several variables that describe it, the more consistent and more reliable results one should get at the outcome of their classifier. This is true in general when one works with datasets, as it is discussed in [22]. For instance, if instead of 400 records belonging to males we had 600 and only 200 for females it is automatic that females are underrepresented, and thus the classifier is biased towards men. Then it is said that the database is not gender-balanced. The same applies to other variables such as age. Let us discuss this further with a curious example.

In [3], a group of researchers claimed to discriminate patients that suffered PD from other kind of neurological diseases (ND group, they called it) with an accuracy of 0.90. Then, a second group of researchers claimed in [17] that they achieved the up to 0.96 accuracy when discriminating PD patients from healthy ones, all using the same approach. The second group of researchers were trying to point out that the first group used a wrong experimental design and thus they were not classifying diseases, but their classifier was classifying something else. That is what happens when the classifier is confused with the so-called confounding variables. These are always present in classification, but can be largely suppressed using a balanced database. What happened to the first group of researchers is that they used a database that was unbalanced and wrongly experimentally designed. As a consequence, their classifier was not detecting PD, but other variables such as the microphones used for recording the signals present in the database they used.

The database used in this work is therefore gender and age-balanced, since these are relevant factors to PD [20]. In figures 4.1 and 4.2, the histograms of the signals distributed by gender and sex are presented.

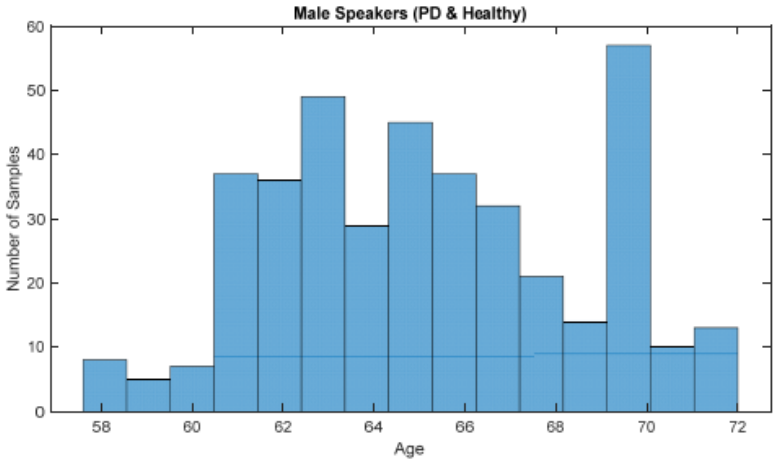

Figure 4.1: Age histogram of male speakers for PD and healthy groups.

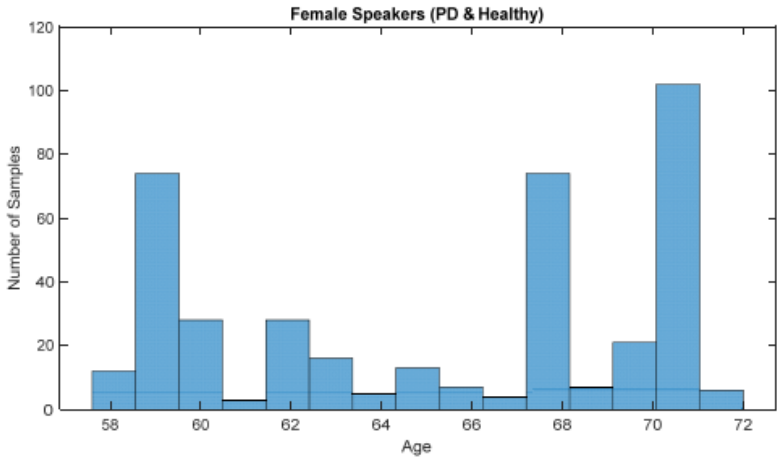

Figure 4.2: Age histogram of female speakers for PD and healthy groups.

It is relevant to note that even these signals are not fully balanced, meaning that the histograms are not fully evenly distributed. Therefore, a bias is acknowledged to be to some extent present. In reality, and for practical reasons, it is very difficult to obtain a fully balanced database in all its variables. However, not that the range of age is 58-60 years old. It is precisely in the third age where most of PD cases occur [20].

4.1.2 Noise Database

For the purposes of creating a multi-SNR model, some noise is added to the clean signals presented in the last section. The database containing the noise used is the Aurora database [14], widely used in speech processing with more than 2,000 citations. The duration of the noise signals is also 10 seconds. They are mono signals sampled at 8kHz, and are distributed in the following categories:

- Airport
- Babble
- Car
- Exhibition
- Restaurant
- Street
- Subway
- Train

4.2 Experimental Settings

This section presents the parameters used throughout the stages of front-end processing and the GMM-UBM model.

4.2.1 Front End Processing

As mentioned in Chapter 3, PLP is the feature extraction method used in the experimental part of this thesis. The signals containing the speech were down-sampled to 8kHz to match the noise's sampling rate. For windowing, a 30ms duration Hamming window with a hop size of 20ms was used. With this windows size and 12 as the linear prediction order, one gets a total of 39 coefficients distributed as follows:

- 12 linear coefficients + its corresponding energy (0th order coefficient) = 13.

- 13 delta coefficients.
- 13 delta-delta coefficients.

The choice of these parameters follows [23].

The resulting PLP matrix for a given signal has thus a size of 39 times roughly 500 (depending on the duration of the signal, which is roughly 10s). In figure 4.3, the spectrogram of the PLP matrix is thereby given:

Figure 4.3: PLP spectrogram of a healthy female speaker at 30dB SNR. Time frames in the X-axis and linear PLP coefficients on the Y-axis.

The computation of the PLP coefficients was done after adding noise to the signals at different SNRs. The SNRs chosen were -10, 0, 10, 20 and 30dB. In the following snippet, the pseudo-code containing the process described in this subsection is presented. The full version of the code can be found in A.1.

```
n_signals = 200;
fs = 8000;
lpc_order = 12;
FLT = 0.030;
FST = 0.020;
noise = {'airport.wav','babble.wav','car.wav','exhibition.wav',...
'restaurant.wav','street.wav','subway.wav','train.wav'};
snr = (-10:10:30);

for all healthy female signals do

---Read signals -> Clean signals
---Downsample signals -> Downsamples Clean Signals

---for all snrs do
```

```
---Choose type of noise randomly
---Add noise at the given snr to the clean signals -> Noisy Signals
---Compute PLPs from Noisy Signals -> PLP Coefficients

---end

end

% Result: (200x5) cell containing PLPs for every signal and SNR.
% Repeat process for healthy males as well as PD male and female groups
```

4.2.2 GMM-UBM Model

The multi-SNR model was created employing a GMM-UBM model for each SNR used in the last subsection, giving a total of 5 models. Each model at the same time was tested on the 5 different SNRs.

For each GMM-UBM model the following parameters were used:

- 32 different mixtures. Following [23].
- 10 number of iterations in the final split of the EM algorithm.
- 5 as the MAP relevance factor.

The process followed in the construction of every GMM-UBM model is described in the following pseudo-code. The full version of the code can be found in A.2.

```
n_mixtures = 2^5;
final_niter = 10;
map_tau = 5;
```

```
training_snr = (-10:10:30);
test_snr = (-10:10:30);
n_iter = 50;

Load PLP coefficients

for all training snr

---for all test snr

------for all iterations

------Define training set randomly for healthy and PD at given training SNR
------Define test set randomly for healthy and PD at given test SNR
------Define test labels

------Create UBM for healthy and PD using training data
------Create adapted models for healthy and PD using training data

------Compare predicted labels of test set with the test set defined labels
------Get results per iteration

------end

---Get mean results and plot

---end

end
```

4.2.3 Evaluation

This subsection covers the evaluation part of the models. A 5-fold cross validation (henceforth CV) was used, following [16]. That is, 0.8 of the data used for training and 0.2 for testing. This process was then iterated 10 times, giving a total of 50 different iterations. In other words, from the 800 signals, 640 were randomly selected for training and 160 were

randomly selected for testing at each of the 50 iterations. Afterwards, the mean of the results was taken.

Accuracy is the metric common used with classification. When classifying between two different classes, the problem is usually called detection. In detection, other metrics are used. One of the most common and useful for its interpretability is the receiver operating characteristic (henceforth ROC) curve. Instead of presenting (only) accuracy, the area under the ROC curve (henceforth AUC) is given.

Let us define these terms properly. After every iteration of the process, the confusion matrix (henceforth CM) is computed. This is a matrix depicted in 4.4:

Confusion Matrix		*Modeled Values*: x_m	
		False	*True*
Actual Values: x	*False*	*True Negatives*	*False Positives*
	True	*False Negatives*	*True Positives*

Figure 4.4: General aspect of a CM. Taken from [39].

where:

- True negative or TN: the true label was healthy and the detector said the person was healthy.
- False positive or FP: the true label was healthy and the detector said the person had PD.
- False negative or FN: the true label was PD and the detector said the person was healthy.
- True positive or TP: the true label was PD and the detector said the person had PD.

Now these are defined, we can define accuracy:

$$\text{accuracy} = \frac{TN + TP}{TN + TP + FP + FN} = \frac{\text{\# of correct assessments}}{\text{total \# of observations}} \tag{4.1}$$

The ROC curve is built plotting the different values for the FP and TP rates. They are often called fall-out and recall and are defined as:

$$\text{FP Rate} = \frac{FP}{FP + TN} \tag{4.2}$$

$$\text{TP Rate} = \frac{TP}{TP + FN} \tag{4.3}$$

In figure 4.5, an example of ROC curve is presented. The AUC is thus the area under the ROC curve, as was aforementioned.

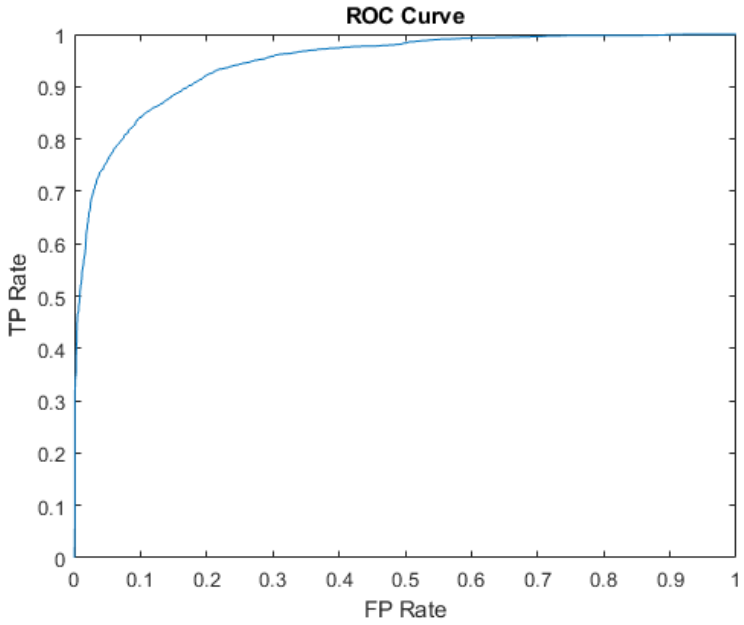

Figure 4.5: ROC curve example. FP and TP rates in the x and the y-axis, respectively.

Throughout Chapter 5, both accuracy and ROC AUC are used as metrics.

Chapter 5

Results

The results gathered in the experimental part of this thesis are presented in this section.

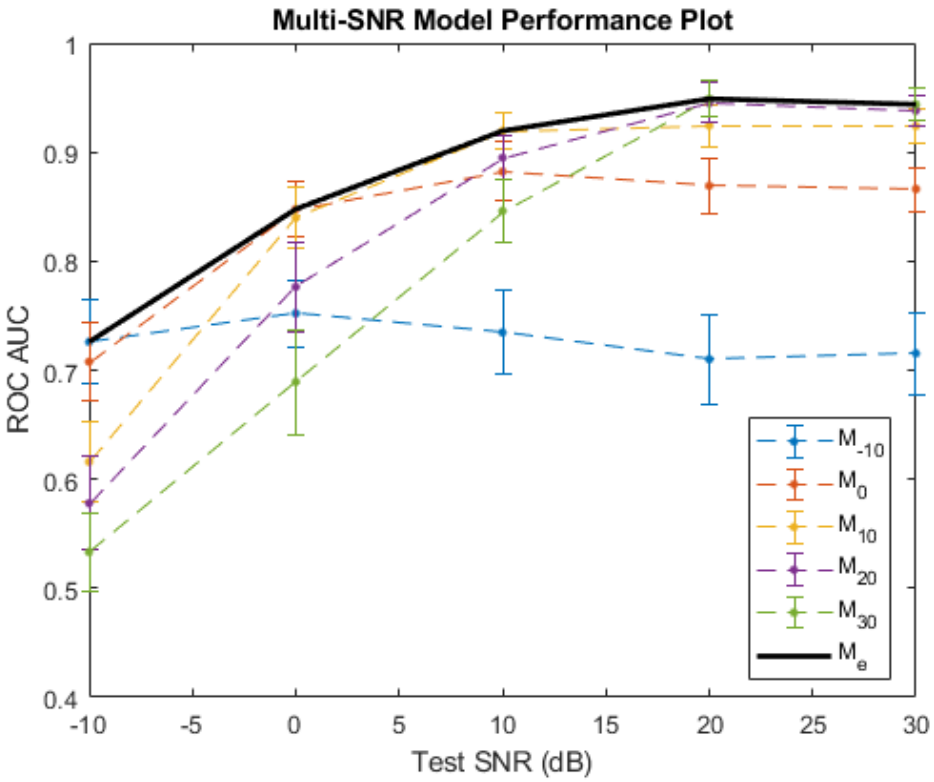

Figure 5.1: Multi-SNR model performance plot. Test SNR vs mean ROC AUC for every training model ($M_{train-SNR}$). The confidence intervals at 68 percent confidence are also provided. The models are represented by dashed lines. All the points present in this lines were computed fitting the original means of the ROC AUC to a 3rd degree polynomial. For a given test SNR, the best ROC AUC is taken and all the values are fit to a 3rd degree polynomial. These values generate the 'enhanced' (M_e) model represented with the solid black line.

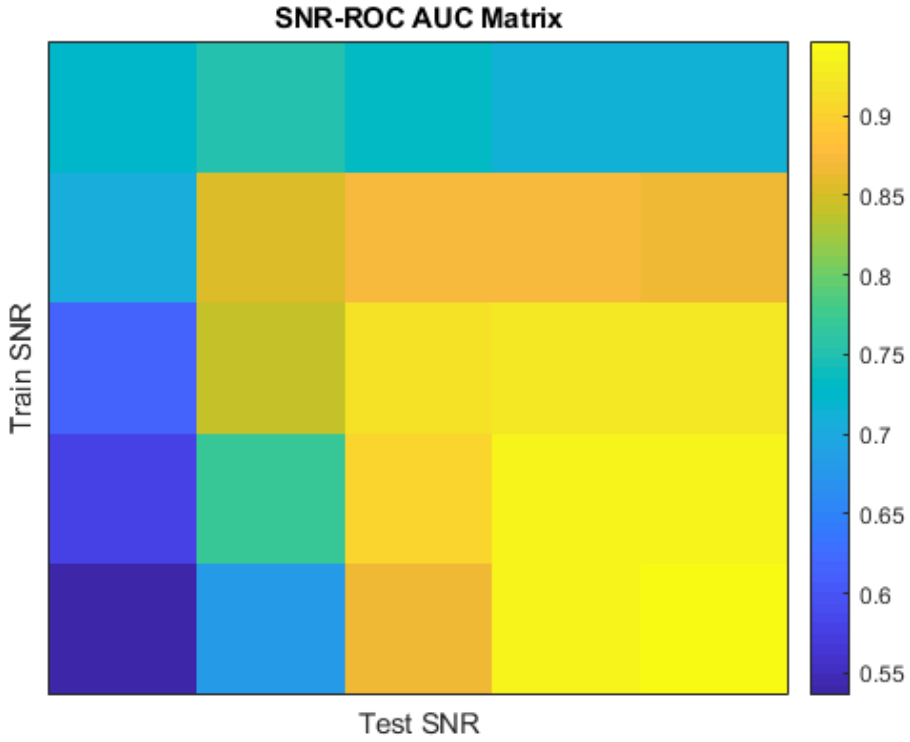

Figure 5.2: Multi-SNR model matrix plot. Test SNR vs Train SNR vs mean ROC AUC. This matrix represents the ROC AUC values as colours. The columns the test SNR values, that is from -10 to 30 in jumps of 10 in decibels. The rows represent the train SNR, which are the same values as the test SNR. Each square thus represents a ROC AUC mean value for a given train and test SNR. The color grading legend is on the right side of the figure.

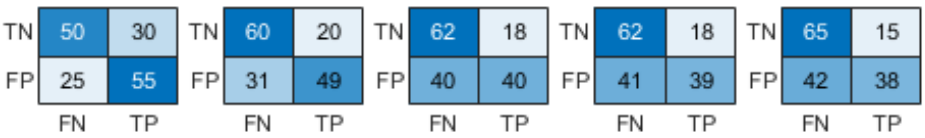

Figure 5.3: Confusion matrices corresponding to the -10dB SNR model. From left to right, evaluated respectively with the different test SNRs: -10, 0, 10, 20 and 30dB.

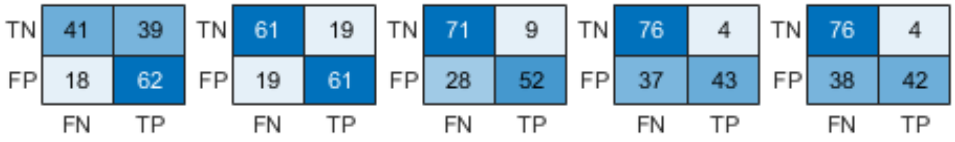

Figure 5.4: Confusion matrices corresponding to the 0dB SNR model. From left to right, evaluated respectively with the different test SNRs: -10, 0, 10, 20 and 30dB.

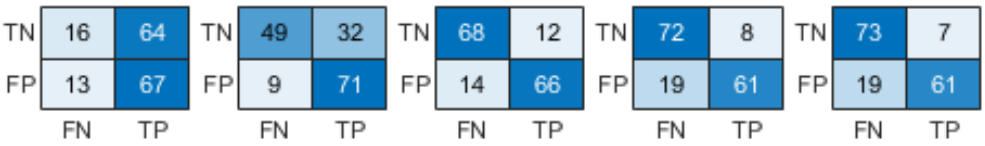

Figure 5.5: Confusion matrices corresponding to the 10dB SNR model. From left to right, evaluated respectively with the different test SNRs: -10, 0, 10, 20 and 30dB.

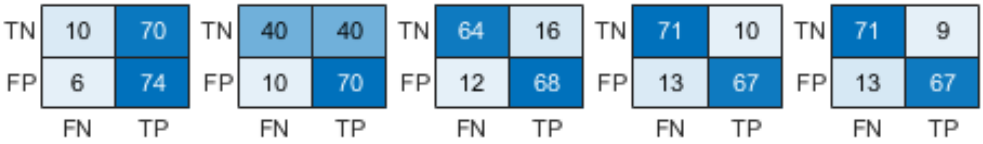

Figure 5.6: Confusion matrices corresponding to the 20dB SNR model. From left to right, evaluated respectively with the different test SNRs: -10, 0, 10, 20 and 30dB.

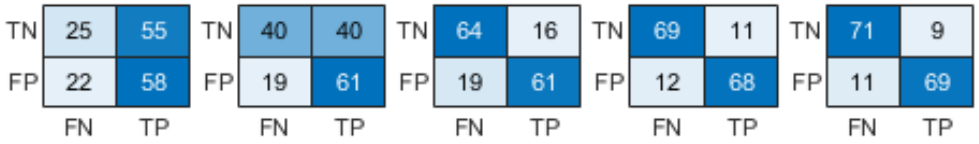

Figure 5.7: Confusion matrices corresponding to the 30dB SNR model. From left to right, evaluated for the different test SNRs: -10, 0, 10, 20 and 30dB.

Chapter 6

Discussion

Let us discuss the results presented in the last chapter. From figure 5.1, that results tend to get better when the test SNR increases. This should not come as a surprise: the less noise present in the signals (both for training and testing) the better the performance should be. This fact is also reflected on the confidence intervals: the grow larger the worse the SNR gets.

It can also be noticed that the larger the mismatch between the train and test SNRs, the better the results are. This proves the hypothesis of this thesis. Let us analyze a specific case. For a given test SNR of -10dB, the model that gives best performance is indeed the -10dB one. It is true that the 0dB model falls in the same statistical group as a t-test reveals. However, what is important is that the models with higher SNRs are far from the first two aforementioned in terms of performance proving thus the hypothesis. It is true that this difference in performance shrinks for higher SNRs, when taking into account the confidence intervals, but the same argument applies when looking at higher test SNRs. This fact could be alleviated performing more iterations in the training-test process of the algorithm, as this would decrease the standard deviation.

Let us now look at figure 5.2. There one can see graphically that when the mismatch between the SNRs increases, performance gets worse. The cases where the mismatch vanishes are represented in the diagonal of the matrix. In the same way, the elements far from the diagonal depict the situation with bigger mismatch. Indeed, the elements in the diagonal are greater that the elements that surround it. Fact that becomes less obvious as both training and test SNR increase, for performance increases with both training and test

SNR. Therefore, this graph proves the same points that figure 5.2 less in detail but perhaps more intuitively.

Let us take a quick look at the confusion matrices presented in figures 5.3-5.7. It should not come as a surprise that one can come to the same conclusions by analyzing the results gathered in these matrices, but it would perhaps take a lot more time. Let us see the general trend the exhibit. We can see that worst results are in general given at lowest SNR (the first matrix in all figures on the left). From these, the best results are present in the first model (-10dB SNR). This we should expect since the mismatch in this case is zero. We can also see that results get better and better when the training and test SNR increase, as pointed out earlier in this chapter.

Chapter 7

Conclusions

In this thesis, a multi-SNR model approach to Parkinson's disease detection through speech analysis was presented. This work was done by means to investigating on the impact of the mismatch in SNR between the training and test data present when using algorithms for PD detection. Different approaches to PD detection were presented. These included approaches at signal, feature, model and score-level. A description of the commonly used feature extraction methods as well as the main PD detection techniques was also given. Details on the GMM-UBM models used were presented as well as a description of PLP coefficients used in the experimental part of the thesis. Specific details on the experimental settings were also given for reproducibility of the results. Finally, the results gathered were presented and later on discussed.

The main hypothesis of this thesis was proven. Not only that, but results are behave according to the effects due to the presence of noise in the signals. That is, the less noise, the better the performance. Speaking of performance, results got as high as 0.95 ± 0.014 ROC AUC or 0.87 ± 0.23 accuracy for somewhat clean speech (30dB) in both training and test SNRs (that is, zero mismatch). On the other hand, in very noisy conditions (-10dB, zero mismatch), results went down to 0.73 ± 0.038 ROC AUC and 0.66 ± 0.035.

For further work, a different feature extraction method could be tested, such as MFCCs, as was done in [23]. Furthermore, to test the multi-SNR model presented in this thesis, an SNR estimator could be employed [2] to estimate the SNR of new data and from that, choose the SNR model that best fits these new data for maximizing performance. An alternative model based on neural networks could also be tested, given their popularity

and good performance.

Bibliography

[1] M. A. Little A. H. Poorjam J. R. Jensen and M. G. Christensen. "A parametric apporach for classification of distortions in pathological voices". In: (2018).

[2] M. A. Little A. H. Poorjam J. R. Jensen and M. G. Christensen. "Supervised Approach to global Signal-to-Noise Ratio Estimation for Whispered and Pathological Voices". In: (2018).

[3] Benba et al. "Discriminating Between Patients With Parkinson's and Neurological Diseases Using Cepstral Analysis". In: (2016).

[4] Bocklet et al. "Automatic Evaluation of Parkinson's Speech - Acoustic, Prosodic and Voice Related Cues". In: (2013).

[5] Bot et al. "The PDQ-8: development and validation of a short-form Parkinson's disease questionnaire". In: (1997).

[6] Dehak et al. "Discriminative and Generative Approches for Long- and Short-Term Speaker Characteristics Modeling: Application to Speaker Verification". In: (2009).

[7] Dehak et al. "Front-End Factor Analysis For Speaker Verification". In: (2010).

[8] Dempster et al. "Maximum likelihood from incomplete data via the EM algorithm". In: (1977).

[9] Drygajlo et al. "Speaker verification in noisy environments with combined spectral subtraction and missing feature theory". In: (1998).

[10] Han et al. "Convolutional neural networks with binaural representations and background subtraction for acoustic scene classification". In: (2017).

[11] Jenkinson et al. “The mPower study, Parkinson disease mobile data collected using ResearchKit”. In: (2016).

[12] Kenny et al. “Joint Factor Analysis versus Eigenchannels in Speaker Recognition”. In: (2007).

[13] Ming et al. “Robust Speaker Recognition in Noisy Conditions”. In: (2007).

[14] Pearce et al. “The Aurora Experimental Framework for the Performance Evaluation of Speech Recognition Systems under Noisy Conditions”. In: (2000).

[15] Reynolds et al. “Speaker Verification Using Adapted Gaussian Mixture Models”. In: (2000).

[16] Rodriguez et al. “Sensitivity Analysis of k-Fold Cross Validation in Prediction Error Estimation”. In: (2010).

[17] Rusz et al. “High-Accuracy Voice-Based Classification Between Patients With Parkinson’s Disease and Other Neurological Diseases May Be an Easy Task With Inappropriate Experimental Design”. In: (2017).

[18] Saastamoinen et al. “On Factors Affecting MFCC-Based speaker Recognition Accuracy”. In: (2005).

[19] Stevens et al. “Critical band width in loudness summation”. In: (1957).

[20] Tanner et al. “Incidence of Parkinson’s Disease: Variation by Age, Gender, and Race/Ethnicity”. In: (2003).

[21] Teunen et al. “A model-based transformational approach to robust speaker recognition”. In: (2000).

[22] Torralba et al. “Unbiased Look at Dataset Bias”. In: (2011).

[23] Velázquez et al. “Analysis of speaker recognition methodologies and the influenceof kinetic changes to automatically detect Parkinson’s Disease”. In: (2017).

[24] Sage Bionetworks. *Sage Bionetworks*. URL: `http://sagebionetworks.org/`.

[25] Bishop. “Pattern recognition and machine learning”. In: (2006).

[26] Heywood Brothers. *Patients Like Me*. URL: `https://www.patientslikeme.com/`.

[27] Dave. “Feature Extraction Methods LPC, PLP and MFCC In Speech Recognition”. In: (2013).

[28] Dinov. “Expectation Maximization and Mixture Modeling Tutorial”. In: (2008).

[29] S. Escanes. “Acoustic scene classification using deep learning techniques”. In: (2018).

[30] Jason Fisher. *Life Insurance and Life Expectancy | Visual Data*. URL: https://www.bestliferates.org/blog/life-expectancy/.

[31] Parkinson’s Foundation. *Parkinson’s Foundation*. URL: https://parkinson.org/understanding-parkinsons/what-is-parkinsons.

[32] Hermansky. “Perceptual linear predictive (PLP) analysis of speech”. In: (1990).

[33] Markel and Gray.

[34] Global Health Metrics. “Global, regional, and national age-sex-specific mortality for 282 causes of death in 195 countries and territories, 1980–2017: a systematic analysis for the Global Burden of Disease Study 2017”. In: (2018).

[35] Orozco-Arroyave. “Automatic detection of Parkinson’s disease in running speech spoken in three different languages”. In: (2016).

[36] Pillay. “Voice biometrics under mismatch noise conditions”. In: (2010).

[37] TED. *Max Little: A test for Parkinson’s with a phone call*. URL: https://www.youtube.com/watch?v=HWsehvUkI-c.

[38] Titze. “Principles of voice production”. In: (1999).

[39] WebMD. *Picture of the Ear*. URL: https://www.webmd.com/cold-and-flu/ear-infection/picture-of-the-ear#1.

[40] Zwicker. “Subdivision of the audible frequency range into critical bands”. In: (1961).

Appendix A

MATLAB Implementation

A.1 PLP CODE

```
clear vars
clc

%% Parameters

fs_desired = 8000;
lpc_order = 12;
RASTA = 0;
FLT = 0.030;
FST = 0.020;
noise = {'airport.wav','babble.wav','car.wav',
'exhibition.wav','restaurant.wav','street.wav','subway.wav','train.wav'};
snr = (-10:10:30);

%% Feature Extraction PD Males

PD_M = dir('C:\Users\stud\Desktop\PD Detection\
PD Detection Audio Files\Clean Signals\PD_M\*.wav');
folders_PD_M = {PD_M.folder}';
```

```
PD_M_files = {PD_M.name}';
total_PD_M_file = length(PD_M_files);
complt_feat_set_PD_M = cell(total_PD_M_file,length(snr));

parfor_progress(total_PD_M_file);

for ix = 1:total_PD_M_file

    file_name = strcat(folders_PD_M{ix},'\',PD_M_files{ix});
    [clean_signal,fs_current] = audioread(file_name);

    if fs_desired ~= fs_current
        clean_signal=resample(clean_signal,fs_desired,fs_current);
    end

    for k = 1:length(snr)
        perm = randperm(8);
        [noisy_signal,newNoise] =
        add_noisem(clean_signal,noise{perm(1)},snr(k),fs_desired);
        noisy_signal = noisy_signal - mean(noisy_signal);
        noisy_signal = noisy_signal/max(abs(noisy_signal));

        speechInd = energy_based_vad(noisy_signal,FLT,FST,fs_desired);

        PLP_Coefficients_speech_frames =

        Compute_VAD_RASTA_PLP(noisy_signal, fs_desired,

        speechInd,lpc_order,FLT,FST,RASTA);

        complt_feat_set_PD_M{ix,k} = PLP_Coefficients_speech_frames;
    end
    parfor_progress;
end
parfor_progress(0);

% save('C:\Users\stud\Desktop\PD Detection\PD Detection PLP Features
\HL_Fn\PD_M_PLP.mat', '-v7.3', 'complt_feat_set_PD_M','PD_M_files')
```

```
%% Feature Extraction PD Females

PD_F = dir('C:\Users\stud\Desktop\PD Detection\
PD Detection Audio Files\Clean Signals\PD_F\*.wav');
folders_PD_F = {PD_F.folder}';
PD_F_files = {PD_F.name}';
total_PD_F_file = length(PD_F_files);
complt_feat_set_PD_F = cell(total_PD_F_file,length(snr));

parfor_progress(total_PD_F_file);

for ix = 1:total_PD_F_file

    file_name = strcat(folders_PD_F{ix},'\',PD_F_files{ix});
    [clean_signal,fs_current] = audioread(file_name);

    if fs_desired ~= fs_current
        clean_signal=resample(clean_signal,fs_desired,fs_current);
    end


        for k = 1:length(snr)

            perm = randperm(8);
            [noisy_signal,newNoise] = add_noisem(clean_signal,noise{perm(1)},snr(k),fs_desired);
            noisy_signal = noisy_signal - mean(noisy_signal);
            noisy_signal = noisy_signal/max(abs(noisy_signal));

            speechInd = energy_based_vad(noisy_signal,FLT,FST,fs_desired);

            PLP_Coefficients_speech_frames =

            Compute_VAD_RASTA_PLP(noisy_signal, fs_desired, speechInd,lpc_order,FLT,FST,RASTA);

            complt_feat_set_PD_F{ix,k} = PLP_Coefficients_speech_frames;
        end

    parfor_progress;
end
```

```
parfor_progress(0);

%% Feature Extraction Healthy Males

HL_M = dir('C:\Users\stud\Desktop\PD Detection\
PD Detection Audio Files\Clean Signals\HL_M\*.wav');
folders_HL_M = {HL_M.folder}';
HL_M_files = {HL_M.name}';
total_HL_M_file = length(HL_M_files);
complt_feat_set_HL_M = cell(total_HL_M_file,length(snr));

parfor_progress(total_HL_M_file);

for ix = 1:total_HL_M_file

    file_name = strcat(folders_HL_M{ix},'\',HL_M_files{ix});
    [clean_signal,fs_current] = audioread(file_name);

    if fs_desired ~= fs_current
        clean_signal=resample(clean_signal,fs_desired,fs_current);
    end


        for k = 1:length(snr)
            perm = randperm(8);
            [noisy_signal,newNoise] =

            add_noisem(clean_signal,noise{perm(1)},snr(k),fs_desired);
            noisy_signal = noisy_signal - mean(noisy_signal);
            noisy_signal = noisy_signal/max(abs(noisy_signal));

            speechInd = energy_based_vad(noisy_signal,FLT,FST,fs_desired);

            PLP_Coefficients_speech_frames =

            Compute_VAD_RASTA_PLP(noisy_signal, fs_desired, speechInd,
            lpc_order,FLT,FST,RASTA);

            complt_feat_set_HL_M{ix,k} = PLP_Coefficients_speech_frames;
```

```
        end

    parfor_progress;
end
parfor_progress(0);

%% Feature Extraction Healthy Females

HL_F = dir('C:\Users\stud\Desktop\PD Detection\
PD Detection Audio Files\Clean Signals\HL_F\*.wav');
folders_HL_F = {HL_F.folder}';
HL_F_files = {HL_F.name}';
total_HL_F_file = length(HL_F_files);
complt_feat_set_HL_F = cell(total_HL_F_file,length(snr));

parfor_progress(total_HL_F_file);

for ix = 1:total_HL_F_file

    file_name = strcat(folders_HL_F{ix},'\',HL_F_files{ix});
    [clean_signal,fs_current] = audioread(file_name);

    if fs_desired ~= fs_current
        clean_signal=resample(clean_signal,fs_desired,fs_current);
    end


        for k = 1:length(snr)
             perm = randperm(8);
            [noisy_signal,newNoise] =

            add_noisem(clean_signal,noise{perm(1)},snr(k),fs_desired);
            noisy_signal = noisy_signal - mean(noisy_signal);
            noisy_signal = noisy_signal/max(abs(noisy_signal));

            speechInd = energy_based_vad(noisy_signal,FLT,FST,fs_desired);

            PLP_Coefficients_speech_frames = Compute_VAD_RASTA_PLP(noisy_signal, fs_desired,
            speechInd,lpc_order,FLT,FST,RASTA);
```

```
            complt_feat_set_HL_F{ix,k} = PLP_Coefficients_speech_frames;
        end

    parfor_progress;
end
parfor_progress(0);
```

A.2 UBM-GMM CODE

```
clear vars
close all
clc

%% Parameters

fs_desired = 8000;
RASTA = 0;
lpc_order = 12;
map_tau = 5;
config = 'mvw'; % 'mwv'
n_mixtures = 2.^5;
nWorkers = 4;
final_niter = 10;
ds_factor = 1;

%% Load data

load('C:\Users\stud\Desktop\New-PLPs\new_HL_F_PLP.mat')
load('C:\Users\stud\Desktop\New-PLPs\new_HL_M_PLP.mat')
load('C:\Users\stud\Desktop\New-PLPs\new_PD_F_PLP.mat')
load('C:\Users\stud\Desktop\New-PLPs\new_PD_M_PLP.mat')

%%
```

```
total_file = size(new_HL_F_PLP,1);
FOLD = 5;
ITR = 10;

tst_lbl = cell(FOLD*ITR,1);
SCORES = cell(FOLD*ITR,1);
auc = zeros(FOLD*ITR,1);
CM = cell(FOLD*ITR,1);
all_train_ind = cell(FOLD*ITR,4);
all_test_ind = cell(FOLD*ITR,4);
all_models = cell(FOLD*ITR,1);
TRAINING_FILES = cell(FOLD*ITR,4);
TEST_FILES = cell(FOLD*ITR,4);
pred_labels = cell(FOLD*ITR,1);
Mean_CM_Percent = cell(5,5);
% mean_acc_matrix = nan(5,5);
STD_CM_Percent = cell(5,5);
% std_acc_matrix = nan(5,5);
mean_auc_matrix = nan(5,5);
std_auc_matrix = nan(5,5);
TP = nan(FOLD*ITR,1);
TN = nan(FOLD*ITR,1);
FP = nan(FOLD*ITR,1);
FN = nan(FOLD*ITR,1);
TP_mean = nan(5,5);
TN_mean = nan(5,5);
FP_mean = nan(5,5);
FN_mean = nan(5,5);
mean_CM = cell(5,5);
acc = nan(FOLD*ITR,1);
mean_acc_matrix = nan(5,5);
std_acc_matrix = nan(5,5);




tic
for m = 1:5
```

```
    figure()
for n = 1:5
        for itx = 1:ITR*FOLD

    train_ind_hl_m = sort(randperm(total_file,(1-1/FOLD)*200));
    train_ind_hl_f = sort(randperm(total_file,(1-1/FOLD)*200));
    train_ind_pd_m = sort(randperm(total_file,(1-1/FOLD)*200));
    train_ind_pd_f = sort(randperm(total_file,(1-1/FOLD)*200));

    TRAINING_FILES{itx,1} = new_HL_M_PLP(train_ind_hl_m);
    TRAINING_FILES{itx,2} = new_HL_F_PLP(train_ind_hl_f);
    TRAINING_FILES{itx,3} = new_PD_M_PLP(train_ind_pd_m);
    TRAINING_FILES{itx,4} = new_PD_F_PLP(train_ind_pd_f);

    all_train_ind{itx,1} = train_ind_hl_m;
    all_train_ind{itx,2} = train_ind_hl_f;
    all_train_ind{itx,3} = train_ind_pd_m;
    all_train_ind{itx,4} = train_ind_pd_f;

    ts_ind_hl_m = sort(setdiff(1:total_file,train_ind_hl_m));
    ts_ind_hl_f = sort(setdiff(1:total_file,train_ind_hl_f));
    ts_ind_pd_m = sort(setdiff(1:total_file,train_ind_pd_m));
    ts_ind_pd_f = sort(setdiff(1:total_file,train_ind_pd_f));

    TEST_FILES{itx,1} = new_HL_M_PLP(ts_ind_hl_m);
    TEST_FILES{itx,2} = new_HL_F_PLP(ts_ind_hl_f);
    TEST_FILES{itx,3} = new_PD_M_PLP(ts_ind_pd_m);
    TEST_FILES{itx,4} = new_PD_F_PLP(ts_ind_pd_f);

    all_test_ind{itx,1} = ts_ind_hl_m;
    all_test_ind{itx,2} = ts_ind_hl_f;
    all_test_ind{itx,3} = ts_ind_pd_m;
    all_test_ind{itx,4} = ts_ind_pd_f;

    train_hl_folds = [new_HL_M_PLP(train_ind_hl_m,:);
    new_HL_F_PLP(train_ind_hl_f,:)];
    % all males & females
%     train_hl_folds = HL_M_PLP(train_ind_hl_m,:); % males
%     train_hl_folds = HL_F_PLP(train_ind_hl_f,:); % females
```

```
    train_pd_folds = [new_PD_M_PLP(train_ind_pd_m,:);
    new_PD_F_PLP(train_ind_pd_f,:)];
    % all males & females
%     train_pd_folds = PD_M_PLP(train_ind_pd_m,:); % males
%     train_pd_folds = PD_F_PLP(train_ind_pd_f,:); % females


    test_hl_folds  = [new_HL_M_PLP(ts_ind_hl_m,:);new_HL_F_PLP(ts_ind_hl_f,:)];
    % all males & females
%     test_hl_folds  = HL_M_PLP(ts_ind_hl_m,:); % males
%     test_hl_folds  = HL_F_PLP(ts_ind_hl_f,:); % females
    test_pd_folds  = [new_PD_M_PLP(ts_ind_pd_m,:);new_PD_F_PLP(ts_ind_pd_f,:)];
    % all males & females
%     test_pd_folds  = PD_M_PLP(ts_ind_pd_m,:); % males
%     test_pd_folds  = PD_F_PLP(ts_ind_pd_f,:); % females

    test_data_folds   = [test_hl_folds;test_pd_folds];
    test_labels_folds =
    [1*ones(1,length(test_hl_folds)) , 2*ones(1,length(test_pd_folds))]; % Healthy = 1; PD = 2;



        train_hl_folds_snr = train_hl_folds(:,m);
        train_pd_folds_snr = train_pd_folds(:,m);

        ubm = gmm_em([train_hl_folds_snr;train_pd_folds_snr], n_mixtures, final_niter,
        ds_factor, nWorkers);

        HL_model = mapAdapt(train_hl_folds_snr, ubm, map_tau, config);
        PD_model = mapAdapt(train_pd_folds_snr, ubm, map_tau, config);
        gmm_models = {HL_model;PD_model};
        all_models{itx,1} = {HL_model;PD_model;ubm};



            test_data_folds_snr = test_data_folds(:,n);
            Scores = zeros(size(test_labels_folds,2),size(gmm_models,1));

            for i = 1:size(test_labels_folds,2)
```

```
                for j = 1:size(gmm_models,1)

                Scores(i,j) =
                mean(compute_llk(test_data_folds_snr{i},gmm_models{j})) -

                mean(compute_llk(test_data_folds_snr{i},ubm));
                end
            end
            [~,predicted_labels_folds] = cellfun(@max,num2cell(Scores,2));
            pred_labels{itx,1} = predicted_labels_folds;
            CM{itx,1} = confusionmat(test_labels_folds,predicted_labels_folds);
            TP(itx,1) = CM{itx,1}(2,2);
            TN(itx,1) = CM{itx,1}(1,1);
            FP(itx,1) = CM{itx,1}(1,2);
            FN(itx,1) = CM{itx,1}(2,1);
            acc(itx,1) = (TP(itx,1)+TN(itx,1))/(TP(itx,1)+TN(itx,1)+FP(itx,1)+FN(itx,1));
            SCORES{itx,1} = Scores(:,2);
            tst_lbl{itx,1} = test_labels_folds;
            [~,~,~,auc(itx,1)] = perfcurve((test_labels_folds(1,:))',Scores(:,2),2);
            fprintf('------- Iteration %d/%d ---------\n',itx,ITR*FOLD)


        end
        toc

TP_mean(m,n) = mean(TP);
TN_mean(m,n) = mean(TN);
FP_mean(m,n) = mean(FP);
FN_mean(m,n) = mean(FN);
mean_CM{m,n} =
[round(TN_mean(m,n)) round(FP_mean(m,n));

round(FN_mean(m,n)) round(TP_mean(m,n))];
mean_acc_matrix(m,n) = mean(acc);
std_acc_matrix(m,n) = std(acc);

mean_auc_matrix(m,n) = mean(auc)
std_auc_matrix(m,n) = std(auc)
```

```
% save('PD_Detection_GMM_UBM_50_itr.mat','TRAINING_FILES',
'TEST_FILES','all_models','SCORES','auc','tst_lbl', '-v7.3')
CM_Percent = cellfun(@(x) round(100*(x./repmat(sum(x,2),1,2))),CM,'UniformOutput',false);
Mean_CM_Percent{m,n} = round(mean(cat(3, CM_Percent{:}), 3));
% mean_acc_matrix(m,n) =
(Mean_CM_Percent{m,n}(2,2)+Mean_CM_Percent{m,n}(1,1))/
(Mean_CM_Percent{m,n}(2,2)+Mean_CM_Percent{m,n}(1,1)+
Mean_CM_Percent{m,n}(2,1)+Mean_CM_Percent{m,n}(2,2));
STD_CM_Percent{m,n} = round(std(cat(3, CM_Percent{:}), [],3));
% std_acc_matrix(m,n) = (STD_CM_Percent{m,n}(2,2)+STD_CM_Percent{m,n}(1,1))/
(STD_CM_Percent{m,n}(2,2)+STD_CM_Percent{m,n}(1,1)+
STD_CM_Percent{m,n}(2,1)+STD_CM_Percent{m,n}(1,2));

[X,Y,T,AUC] = perfcurve(tst_lbl,SCORES,2,'XVals','All');


% subplot(5,5,n)
plot(X,Y(:,1))
hold on
plot(X,Y(:,2))
plot(X,Y(:,3))
inBetween = [Y(:,2); flipud(Y(:,3))];
fill([X; flipud(X)], inBetween, 'b','FaceAlpha',0.2);
xlabel('FPR')
ylabel('TPR')
ylim([0,1.01])
axis image
savefig('ROC.fig')

% subplot(122)
% imagesc(Mean_CM_Percent),colormap('gray'), axis image
% MCMPT = Mean_CM_Percent';
% text(1,2,[num2str(MCMPT(1,2)),' %'],'FontSize',18,'Color','y')
% text(1,1,[num2str(MCMPT(1,1)),

' %'],'FontSize',18),text(2,1,[num2str(MCMPT(2,1)),' %'],'FontSize',18,'Color','y')
% text(2,2,[num2str(MCMPT(2,2)),' %'],'FontSize',18)
% xticks([1 2]), yticks([1 2])
```

```
% xticklabels({'Predicted Healthy','Predicted PD'})
% yticklabels({'True Healthy','True PD'});


subplot(5,5,n)
xvalues = {'FN','TP'};
yvalues = {'TN','FP'};
heatmap(xvalues,yvalues,mean_CM{m,n},'FontSize', 8,'ColorbarVisible','off')
disp('Done!')
savefig(['CM(' num2str(m) ').fig'])
% saveas(gcf,'CM.png')

end

end
```