Anexos

A. Spectral mapping methods for voice conversion

A.1. GMM-based mapping

A *Gaussian Mixture Model* (GMM) is a statistical distribution constituted by a weighted sum of M simple gaussian distributions, so that the density function of this distribution is:

$$f_x(x) = \sum_{m=1}^{M} \omega_m \mathcal{N}(x; \mu_m; \Sigma_m) \tag{A.1}$$

where ω_m represents the prior probability of the mth gaussian, and $\mathcal{N}(x; \mu_m; \Sigma_m)$ denotes the $D-$dimensional multivariate normal distribution with mean μ_i and covariance matrix Σ_i:

$$\mathcal{N}(x; \mu_m; \Sigma_m) = \frac{1}{(2\pi)^{D/2}\sqrt{|\Sigma_m|}} \exp\left[-\frac{1}{2}(x-\mu_m)^T \Sigma_m^{-1}(x-\mu_m)\right] \tag{A.2}$$

GMM based mapping functions are based on the assumption that the probability density function (PDF) of both source and target features is a GMM distribution. The assumption is reasonable since these distributions with enough number of gaussians can approximate any probability distribution [Fel66].

A.1.1. Source GMM

The GMM approach was first proposed in [Sty98]. In this approach a GMM was built for the source feature vectors, and the mapping function is assumed to be linear for each gaussian in the form:

$$\hat{y}_t = F(x_t) = \sum_{m=1}^{M} P(m|x_t)(A_m x_t + b_m) \tag{A.3}$$

Where $\hat{y}_t$ is the estimated taget feature, the regression matrix A_m and the bias term b_m are to be estimated with *minimum mean squared error* (MMSE) criterion, and the posterior probability of the mth gaussian can be calculated using the Bayes theorem:

$$P(m|x_t) = \frac{\omega_m \mathcal{N}(x_t; \mu_m; \Sigma_m)}{\sum_{m=1}^{M} \omega_m \mathcal{N}(x_t; \mu_m; \Sigma_m)} \tag{A.4}$$

A.1.2. Joint density GMM

The required matrix to obtain the MMSE solution for the source GMM approach is very large and sometimes poorly conditioned, so in [Kai98] it was proposed to build a joint GMM for the source vectors and the target vectors. The source features are augmented with their corresponding target features as $z_t = \left[x_t^T, y_t^T\right]^T$, and $\{z_t\}_{t=1}^N$ modeled with a GMM, such that:

$$f_z(z) = \sum_{m=1}^{M} \omega_m \mathcal{N}(z; \mu_m^{(z)}; \Sigma_m^{(z)}) \tag{A.5}$$

where

$$\mu_m^{(z)} = \begin{bmatrix} \mu_m^{(x)} \\ \mu_m^{(y)} \end{bmatrix} \qquad \Sigma_m^{(z)} = \begin{bmatrix} \Sigma_m^{(xx)} & \Sigma_m^{(xy)} \\ \Sigma_m^{(yx)} & \Sigma_m^{(yy)} \end{bmatrix} \tag{A.6}$$

Under this assumption, x_t and y_t are jointly multivariate gaussian for each gaussian in the GMM, and the conditional distribution probabilities are therefore a GMM as well [Mar79]. The conversion function with MMSE criterion is the expected value of the conditional distribution of y_t given x_t, which is:

$$\hat{y}_t = E\{y_t|x_t\} = \sum_{m=1}^{M} P(m|x_t)\overline{\mu}_m(x_t) = \sum_{m=1}^{M} P(m|x_t)\left[\mu_m^{(y)} + \Sigma_m^{(yx)}\Sigma_m^{(xx)^{-1}}(x_t - \mu_m^{(x)})\right] \tag{A.7}$$

A.1.3. Problems with the GMM-based mapping

These GMM based techniques suffer from several disadvantages inherited from the machine learning models on which they are based. When determining the complexity of the model, there is a trade-off between the generalization capability that it has and the fidelity to the training data.

The most common problems found in GMM approaches are the following:

- Oversmoothing: Simple models tend to remove the fine details of the spectrum and to broaden the formants.
- Overfitting: Models that are too complex may be overfitted to the training data and the generalization capability is therefore low.
- Time-independency: The temporal correlation of the converted features is ignored in these models.

In order to solve these issues, several solutions have been proposed, such as in [Hua01], where post-filtering is introduced to improve oversmoothed spectra. In [Tod07], a maximum-likelihood (ML) estimation of the parameter trajectory is proposed, which partially solves the time independency of the mapping. Additionally, this work proposes using the global variance of the target features to palliate the effect of oversmoothing, which is simpler than the post-filtering method. In [Hel12a], PLS regression is proposed as an alternative to the MMSE solution to reduce the overfitting of the models.

A.2. Nonlinear mapping: Dynamic Kernel Partial Least Squares Regression

Nonlinear mapping can be accomplished using a broad range of techniques, such as artificial neural networks (ANN) [Nar95; Des10], or support vector regression [Son11].

A major drawback of these nonlinear regression systems is the extensive parameter tuning they require. To overcome this problem, in [Hel12b] a new technique called *dynamic kernel partial least squares* (DKPLS) is introduced.

The KPLS method was first introduced in [Ros01], and has the advantage of being able to model nonlinear relations between variables by using a kernel transformation as a preprocessing step. In the DKPLS technique, the dynamics are modeled by augmenting the regressors of every frame with the kernel data from consecutive frames. The regression is performed using PLS criterion, which is able to handle the multicollinearity generated by the use of kernels and the dynamic modeling.

The training scheme for the DKPLS system is depicted in the figure A.1. After the alignment step, as explained in section 4.1, the *k-means* clustering algorithm is applied on the source data to find a set of C reference vectors, and then the feature vectors are non linearly mapped to a higher dimension space via the kernel transformation, as explained in section A.2.1, and the transformed vectors are centered, as the regression model requires the data to be zero-mean. The temporal continuity is taken into account by concatenating the kernel transformed vectors of consecutive frames. Finally, the conversion parameters are found using the PLS method [Jon93].

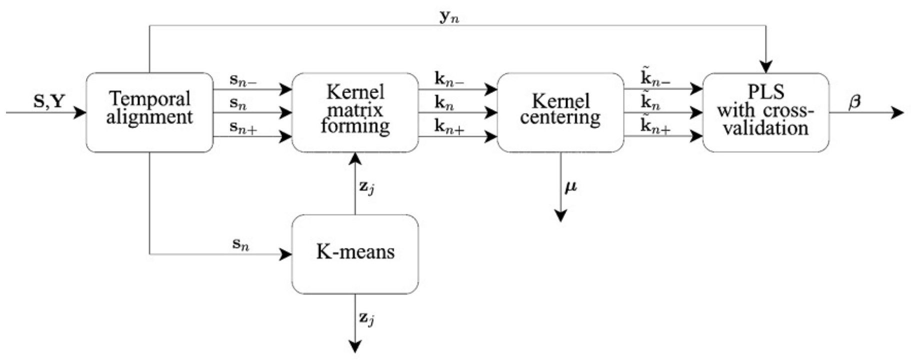


Figura A.1: DKPLS block diagram for training [Hel12b]

A.2.1. Kernel transformation

The kernel transformation is performed using a Gaussian radial basis function kernel as the transformation function. The transformation calculates the similarity between each source vector $\mathbf{s}_n$, $n = 1, \ldots, N$, and the reference vectors obtained from the k-means clustering $\mathbf{z}_j$, $j = 1, \ldots, C$. The Gaussian kernel is defined as

$$k_{jn} = e^{\frac{-\left\|\mathbf{s}_n - \mathbf{z}_j\right\|^2}{2\sigma^2}} \tag{A.8}$$

where σ is the width parameter of the kernel. The selection of σ is not highly crucial, usually it is enough to find a decent range for it.

The transformed vectors are stored into a matrix $\mathbf{K}$ which has the following form:

$$\mathbf{K} = \begin{bmatrix} k_{11} & k_{12} & \ldots & k_{1N} \\ k_{21} & k_{22} & \ldots & k_{2N} \\ \vdots & \vdots & & \vdots \\ k_{C1} & k_{C2} & \ldots & k_{CN} \end{bmatrix} \tag{A.9}$$

To force the bias term of the conversion model to zero, kernel centering is required. Centering in the kernel space is not as obvious as in the original feature space, since the mean cannot be computed directly. For the kernel matrix $\mathbf{K}$, the following steps are applied [Ben03]:

1. Calculate the average of each row in the kernel matrix and subtract the values from $\mathbf{K}$. The averages of the rows are stored into vector μ.

2. Calculate the average of each column and subtract them from $\mathbf{K}$ resulting from Step 1. The resulting column and row-wise centered kernel matrix is denoted as $\tilde{\mathbf{K}}$.

In the conversion process, an analogous procedure is followed, with the exception that the vector which is subtracted from the columns of $\mathbf{K}$ is the μ vector obtained in training.

A.2.2. Dynamic Modeling

In order to take into account the time continuity of the speech features, the DKPLS algorithm relies on augmenting the kernel transformed vectors with its respective previous and following vectors, obtaining a vector x_n as:

$$x_n = \begin{bmatrix} \tilde{\mathbf{k}}_{\mathbf{n}-} \\ \tilde{\mathbf{k}}_{\mathbf{n}} \\ \tilde{\mathbf{k}}_{\mathbf{n}+} \end{bmatrix} \tag{A.10}$$

where $\tilde{\mathbf{k}}_{\mathbf{n}}$ denotes the centered kernel vector for feature vector $\mathbf{s}_{\mathbf{n}}$, and $\tilde{\mathbf{k}}_{\mathbf{n}-}$ and $\tilde{\mathbf{k}}_{\mathbf{n}+}$ the centered kernel vectors for the preceding and following frames of $\mathbf{s}_{\mathbf{n}}$, respectively.

The notation $n+$ and $n-$ is introduced due to the fact that in the training process, the frames are aligned using DTW, which alters the natural ordering of the sequences. However, in conversion the order is unchanged and every frame is processed, and thus the preceding and following frames are the frames $n-1$ and $n+1$ respectively.

A.2.3. Kernel Partial Least Squares

After the kernel transformation, a linear regression model $\mathbf{y}_n = \beta\mathbf{x}_n + \mathbf{e}_n$ is applied, where $\mathbf{y}_n$ denote the unprocessed target vector, $\mathbf{x}_n$ represents the augmented kernel transformed vector of the source feature vector (as in equation A.10), and $\mathbf{e}_n$ represents the regression residual. The entries of $\mathbf{x}_n$ are highly linearly dependent on each other, due to the use of kernels that increases the dimensionality, and to the concatenation of the consecutive transformed vectors.

PLS is a linear regression method that models relationships between a regressor matrix $\mathbf{X}$ and a response matrix $\mathbf{Y}$ which is different from the standard multivariate regression (based on MMSE) in a way that it can cope with collinear data and cases where the number of observations is lower than the number of explanatory variables.

PLS works similarly to principal component analysis (PCA), but in PCA, the principal components are determined by only $\mathbf{X}$ whereas in PLS, both $\mathbf{X}$ and $\mathbf{Y}$ are used for extracting new regressor variables. The aim of PLS is to extract components that capture most of the information in the $\mathbf{X}$ variables that is also useful for predicting $\mathbf{Y}$.

To perform the regression task, PLS constructs new explanatory variables, called *latent variables* or *components*, where each component is a linear combination of $\mathbf{x}_n$, then standard regression methods are used to determine the latent variables in $\mathbf{y}_n$. The number of latent components has to be set beforehand, and the optimal number can be estimated using *cross-validation*.

There exist many algorithms for the PLS regression problem. In this thesis, the SIMPLS algorithm proposed by [Jon93] is used to find the regression matrix β. The SIMPLS algorithm is computationally efficient and avoids calculating matrix inverses.

B. Mean-scale transformation as a PDF equalization

Let $F_0(n)$ be a discrete white Gaussian process with mean μ_x and variance σ_x^2. This assumption implies that each sample x_n is uncorrelated with each other and they follow this Gaussian distribution:

$$f_{x_n}(x_n) = \mathcal{N}(\mu_x, \sigma_x) = \frac{1}{\sqrt{2\pi\sigma_x^2}} \exp\left[-\frac{(x_n - \mu_x)^2}{2\sigma_x^2}\right]$$

The function T that transforms a sample to fit a different PDF is:

$$y_n = T(x_n) = F_y^{-1}(F_x(x_n))$$

where $F_y(\cdot)$ represents the cumulative density function of the target PDF and $F_x(\cdot)$ represents the cumulative density function of the Gaussian PDF. If the target PDF is a Gaussian distribution with mean μ_y and variance σ_y^2, then:

$$F_y(y_n) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \int_{-\infty}^{y_n} e^{\frac{(t-\mu_y)^2}{2\sigma_y^2}} dt = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\frac{y_n-\mu_y}{\sigma_y}} e^{-\frac{t^2}{2}} dt = \Phi\left(\frac{y_n - \mu_y}{\sigma_y}\right)$$
$$F_x(x_n) = \Phi\left(\frac{x_n - \mu_x}{\sigma_x}\right)$$

And therefore:

$$F_y^{-1}(\alpha) = \sigma_y \Phi^{-1}(\alpha) + \mu_y \Longrightarrow F_y^{-1}\left[\Phi\left(\frac{x - \mu_x}{\sigma_x}\right)\right] = \sigma_y \Phi^{-1}\left[\Phi\left(\frac{x - \mu_x}{\sigma_x}\right)\right] + \mu_y$$

So the final transformation function for the sample x_n is:

$$y_n = F_y^{-1}(F_x(x_n)) = \mu_y + \frac{\sigma_y}{\sigma_x}(x_n - \mu_x)$$

Which is the mean-variance scaling transformation.

C. STRAIGHT Framework

STRAIGHT is the most established of the most widely used vocoding methods. It was originally proposed by Kawahara in [Kaw97], and it has been under continuous research and development.

STRAIGHT is a *multi-band mixed excitation vocoder*, which means that it uses additional parameters along with the F_0 value to generate the excitation signal, instead of a regular periodic train of impulses (to reduce the "robotic" effect that they create). STRAIGHT was originally designed as a tool for speech transformation and accurate spectral envelope representation. Original STRAIGHT parameters are represented as Fourier transform magnitudes and aperiodicity measurements corresponding to them.

First, STRAIGHT uses a F_0 extraction algorithm called TEMPO (Time-domain Excitation extractor using Minimum Perturbation Operator). The TEMPO algorithm is based on the following almost harmonic representation of speech:

$$x(t) = \sum_{k=1}^{N} a_k(t) cos \left(\int_0^t \left(k\omega_0(\tau) + \omega_k(\tau) \right) d\tau + \phi_k \right)$$

where $a_k(t)$ represents a slowly time-varying instantaneous amplitude, $\omega_0(\tau)$ is the instantaneous frequency (to estimate), and $\omega_k(\tau)$ is a slowly varying FM component of the kth harmonic.

The fundamental frequency is then extracted by means of a continuous wavelet transform with a Gabor mother wavelet, and calculating a measure called *fundamentalness* [Kaw99], defined as:

$$\begin{aligned} M(t,\tau_c) = &- \log \left[\int_{\Omega_c} \left(\frac{d|D(t,u)|}{du} \right)^2 du \right] + \log \left[\int_{\Omega_c} |D(t,u)|^2 du \right] \\ &- \log \left[\int_{\Omega_c} \left(\frac{d^2 \arg(D(t,u))}{du^2} \right)^2 du \right] + 2 \log(\tau_c) \end{aligned}$$

where $D(t, \tau_0)$ is the wavelet transform of the speech signal, and Ω_c is an integration interval proportional to the size of the analyzing wavelet at scale τ_c. Extracting F_0 is performed by simply finding the maximum in terms of the scale τ_0. The instantaneous frequency is then calculated for each channel τ_c as:

$$\omega_0(t, \tau_c) = \frac{d(\arg(D(t, \tau_c))}{dt}$$

And selecting the fundamental frequency is reduced to finding the maximum of $M(t, \tau_c)$ in terms of τ_c, and then either choose the instantaneous frequency corresponding to that scale, or interpolate through the scales proportionally to the value of M.

Then, the spectral envelope of the signal is calculated by smoothing the spectrum using F_0 adaptive windows with equivalent temporal and spectral resolution. The signal is windowed using two complementary windows:

$$w(t) = e^{-\pi\left(\frac{t}{t_0}\right)^2} h\left(\frac{t}{t_0}\right)$$
$$w_c(t) = w(t) \sin\left(\pi \frac{t}{t_0}\right)$$

where t_0 is the fundamental period, and $h(t)$ is the 2nd order cardinal B-spline function, defined as:

$$h(t) = \begin{cases} 1 - |t| & |t| < 1 \\ 0 & \text{otherwise} \end{cases}$$

Smoothing in the frequency domain is achieved through the 2^{nd} order cardinal B-spline function, which robustly interpolates the sampled spectrum (due to the periodicity of the signal). The complimentary window function $\omega_c(t)$ is sinusoidally modulated so that the spectrogram produces maxima there where the original spectrogram has minima [Kaw99].

The final spectrogram is obtained through the combination of the smoothed spectograms using the window functions:

$$P(\omega, t) = \sqrt{P_0(\omega, t)^2 + \xi P_c(\omega, t)^2}$$

Where ξ is a blending factor that minimizes the temporal variation of the spectogram.

Additionally, STRAIGHT provides measures of aperiodicity for each frequency. The measures are taken by comparing the ratio of the upper and lower spectral envelopes for each frequency, and taking a value from a look-up-table. Then, the aperiodicity values are smoothed by averaging throughout the speech spectrum:

$$P_{AP}(\omega, t) = \frac{\int_{-\infty}^{\infty} w(\lambda, \omega) S(\lambda)^2 \mathrm{LUT}(\lambda) d\lambda}{\int_{-\infty}^{\infty} w(\lambda, \omega) S(\lambda)^2 d\lambda}$$

Where w is a simplified auditory filter centered at frequency ω, and $S(\lambda)^2$ is the speech power spectrum.

D. Classification and Regression Trees

Classification and regression trees (CART) [Bre84] are machine learning tools that are used to predict the outcome of a variable. The goal of a CART is to predict the final value of this variable based on the known values of a certain number of other variables called *predictors*. If the predicted variable has a finite set of possible values (*classes*), the task is then classification, otherwise, the CART performs regression.

The CART addresses the problem by creating recursive binary partitions of the input data. It proceeds by finding the split that minimizes a certain measure of *impurity*. For each possible split based on the values of the predictors, the impurity is calculated, and the data is split in two subsets using the question that minimizes the impurity. The process is repeated for every subset recursively. This splitting sequence constitutes a binary tree structure, where each split can be regarded as a *node*, and the original set can be represented as the *root node*, as illustrated in figure D.1.

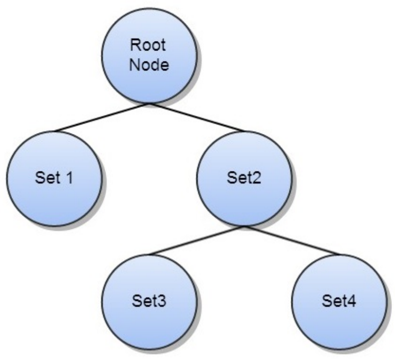


Figura D.1: Split sets as a binary tree

In classification tasks, the impurity measure is usually calculated as the *entropy* of the split data:

$$I(S_t) = -\sum_{x \in X} p(x) \log_2(p(x))$$

where S_t is the subset that would remain after the split at node t, X is the set of classes in S_t, and $p(x)$ is the proportion of elements in class x related to the number of elements in S_t. From this definition, it is clear that when the classification is perfect (i.e, all the elements in S_t belong to the same class), the impurity of the subset is null.

On the other hand, if the purpose is to perform regression, the impurity measure is defined as the MSE of the split subset:

$$I(S_t) = \frac{1}{N_t}\sum_{k=1}^{N_t}(y_k - \mu_t)^2$$

where N_t is the number of elements in the subset S_t, $\{y_k\}$ are the values of the dependent variable at node t, and μ_t represents the mean of the dependent variable at that node.

The size of the tree is an important issue, as very large trees can fit very well the training data, but generalize poorly to new data (known as the *bias-variance dilemma*). In order to determine the optimal size of the tree, a set of sub-trees of different sizes are built by means of *pruning*. The full tree (i.e. that in which no more splits are possible) is pruned by removing the node that provides the lowest change in purity, and the process is repeated recursively, generating a set of pruned sub-trees.

In order to select the optimal sub-tree, first, a cost function called *error-complexity measure* for the depth of the tree is established [Bre84]:

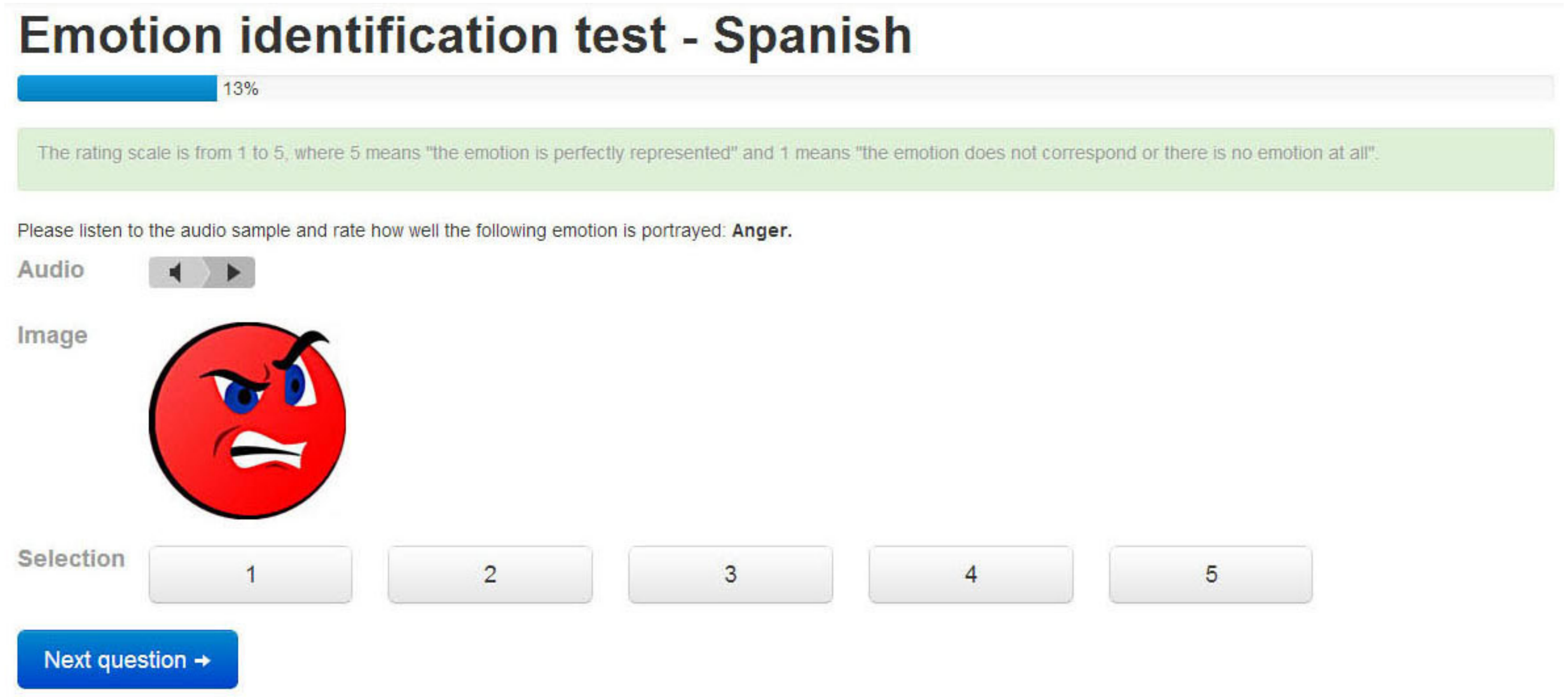
$$EC(T) = MSE(T) + \alpha L$$

where $MSE(T)$ denotes the mean squared error of the tree T, and L denotes number of leaves of the tree. The parameter α is free choice, and for each α, there is a pruned sub-tree that minimizes the error complexity measure.

The optimal parameter α is selected using *cross validation*. The training dataset is divided into M different subsets, and the optimal EC trees (varying α) are built for the $M-1$ training sets. Then, the optimal α is chosen as the one that minimizes the MSE in the testing set.

Once α is selected, a tree is grown from the complete dataset, in a way that minimizes the error-complexity measure using the selected parameter.

E. Example of listening test query



Emotion identification test - Spanish

13%

The rating scale is from 1 to 5, where 5 means "the emotion is perfectly represented" and 1 means "the emotion does not correspond or there is no emotion at all".

Please listen to the audio sample and rate how well the following emotion is portrayed: **Anger.**

Figura E.1: Example of listening test question, querying for the emotion “Anger”. The image has been rotated to enhance resolution.