University of Padova

Source Coding Final Project

LBG Algorithm for CD-Quality Audio Signals

Student:

Umberto Michieli (1150780)

Instructor:

Giancarlo Calvagno

Abstract

Generally speaking the aim of source coding is to compress the original data in a more efficient way in order to store or transmit them more effectively. Many algorithms have been developed in order to represent the information in a more compact form dealing with the two main constraints on storage and transmission capacities.

This report explains the implementation's steps of the LBG algorithm applied to CD-quality audio signals and shows an evaluation of the performances of this technique.

July 17, 2017

1 Introduction

The whole class of compression techniques can be split into two paradigms:

- *Lossless coding:* these techniques do not involve loss of information hence are invertible (i. e. the original data can be exactly recovered from the compressed data). They can be applied only to digital sources, because it is not possible to convert analog signals into digital ones without loss of information, and the basic idea is to exploit variable-length coding in order to compress the data as much as possible mapping most probable symbols into shorter codewords and least probable symbols into longer codewords. The main application of paramount importance is the text compression. Some examples of lossless algorithms are Huffman coding, Arithmetic coding, LZ77, LZ78, LZW and so on. The compression ratio achievable is generally quite small. The *compressed size* is the only parameter that these techniques try to minimize.

- *Lossy coding:* these techniques do involve some loss of information and the resulting coding procedure is generally not invertible (i. e. the original data can not be exactly recovered from the compressed data) and they introduce some distortion in the reconstruction. Furthermore, they are required in case of analog source and can achieve a very high compression ratio, in particular can be significative higher than the lossless coding. These techniques have two different targets: the *compressed size* and the *quality*. In the first case they try to find the minimum possible rate for a given distortion (maximum compression criteria), in the second case they try to minimize the distortion given the rate (maximum fidelity criteria). Some examples of lossy algorithms are the MP3, MPEG, 3GPP and so on.
  Moreover the main idea of the lossy coding is quantization, i. e. a process able to represent a large set (possibly infinite) with a much smaller set. In general quantization consists of two maps: the *encoding function* which maps every symbol belonging to a particular interval into the same codeword (for this reason is generally a not invertible map) and the *decoding function* which generates a reconstruction value that best represents (often in least squares sense) all the values in the interval. Quantization can be performed in two different ways:

  - *Scalar quantization:* where each input sample is quantized independently from the other ones. The most important scalar quantizer is the uniform one.
  - *Vector quantization:* where a block (i. e. a vector) of data is quantized jointly. Here longer sequences of input samples are taken in order to improve the performances (meaning either achieving a lower distortion for a given rate, or a lower rate for a given distortion). A more detailed explanation is provided throughout the next Chapter.

The remainder of the report is organized as follows. Chapter 2 goes deeper into the vector quantization showing the encoding and decoding procedure, then it explains the LBG algorithm and its Matlab implementation. Chapter 3 shows and discusses the main results achieved. Chapter 4 concludes the report with some further observations.

2 Technical Approach

2.1 Vector Quantization

As already mentioned, Vector Quantization (VQ) wants to represent each source sequence with one element belonging to a pre-built representative set of sequences.
VQ picks in input a block of L consecutive source symbols:

$$\text{a vector } \mathbf{x} \in \mathbb{R}^L \text{ where } \mathbf{x} = (x_1, ..., x_L)$$

and outputs an element of the codebook

$$C = \{\mathbf{y}_1, ..., \mathbf{y}_K\}$$

where $\mathbf{y}_i \in \mathbb{R}^L$ with i=1,...,K are called codevectors. A complete definition of a VQ also requires to define a set of decision cells $\{I_i\}_{i=1}^K$ as a partition of $\mathbb{R}^L$ and a quantization rule i. e.:

$$I_i \subseteq \mathbb{R}^L,\ i = 1, ..., K \text{ such that } I_i \cap I_j = \emptyset\ \forall\ i \neq j \text{ and } \bigcup_{i=1}^{K} I_i = \mathbb{R}^L$$

$$Q(\mathbf{x}) = \mathbf{y}_i \text{ if } \mathbf{x} \in I_i$$

Moreover we define the *bitrate* R as the average number of bits per input sample needed to inform the decoder which codevector was selected (see more details in sections 2.1.1 and 2.1.2):

$$R = \frac{1}{L}\lceil \log_2 K \rceil \text{ bits per component}$$

and we measure the per-component distortion D as the mean squared error or the SNR:

$$D = \frac{1}{L}\mathbb{E}[d(\mathbf{x}, Q(\mathbf{x}))] = \frac{1}{L}\int_{\mathbb{R}^L} ||\mathbf{x} - Q(\mathbf{x})||_2^2 f_X(\mathbf{x})d\mathbf{x} = \frac{1}{L}\sum_{i=1}^{K}\int_{I_i} ||\mathbf{x} - \mathbf{y_i}||_2^2 f_X(\mathbf{x})d\mathbf{x}$$

$$\text{SNR} = \frac{\sigma^2_{signal}}{\sigma^2_{noise}}, \qquad \text{SNR}_{\text{dB}} = 10 \cdot \log_{10}(\text{SNR}).$$

Sometimes we can refer to the total distortion which is defined as $D_{tot} = LD$.

2.1.1 Encoding Procedure

At the encoder we must have a set of K L-dimensional vectors (i. e. the codebook), then each codevector is assigned to a binary index. The encoder compares the input vector to each codevector in order to find the one

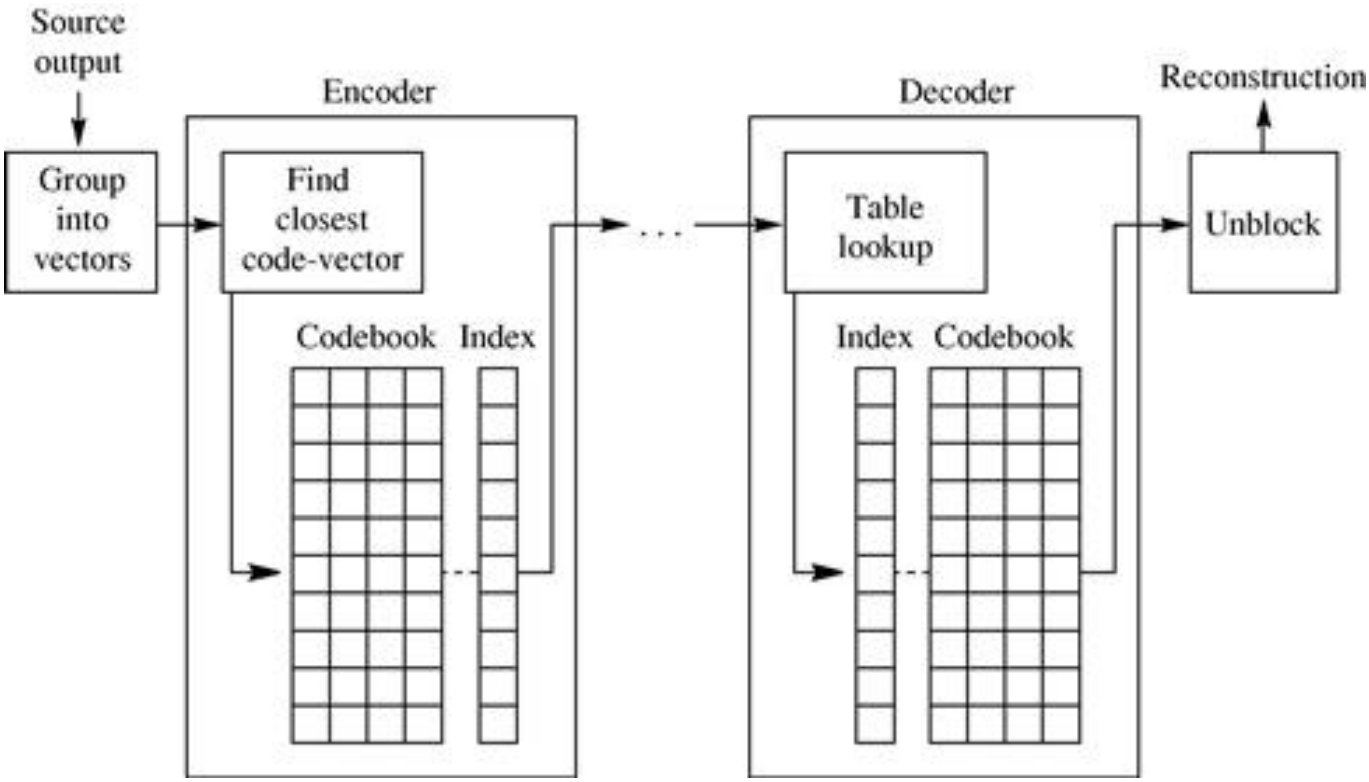

Figure 1: Encoding and decoding scheme of VQ, figure taken from [1]

closest to the input vector (in a least squares sense); then it informs the decoder transmitting the binary index of the selected codevector.

Notice that here the amount of computations is quite large and grows linearly with the size of the codebook.

2.1.2 Decoding Procedure

Also at the decoder we must have the same codebook and each codevector must be assigned to the same binary number as at the encoder. Hence the decoder can retrieve the codevector given its binary number simply consulting a lookup table (much more effective than the operations at the encoder, which makes the decoding procedure lightweight enough to be performed on mobile devices).

2.2 LBG

We can guess that one of the most important design-choice of a VQ is the generation of the codebook. There are two necessary conditions for optimality which are:

- **Nearest Neighbour Condition:** given the set of codevectors $\mathbf{y}_i \in \mathbb{R}^L$ for $i = 1, ..., K$ the optimal partition of $\mathbb{R}^L$ is given by the so-called *Voronoi* decomposition:

$$I_i = \{\mathbf{x} \in \mathbb{R}^L \text{ such that } ||\mathbf{x} - \mathbf{y}_i||_2^2 \leq ||\mathbf{x} - \mathbf{y}_j||_2^2,\ i \neq j\},\ i = 1, ..., K$$

- **Centroid Condition:** given a partition $\{I_i\}_{i=1}^K$ the set of codevectors that minimize the distortion is given by:

$$\mathbf{y}_i = \frac{\int_{I_i} \mathbf{x} f_X(\mathbf{x}) d\mathbf{x}}{\int_{I_i} f_X(\mathbf{x}) d\mathbf{x}} = \int_{I_i} \mathbf{x} f_{X|\mathbf{x}\in I_i}(\mathbf{x}|\mathbf{x} \in I_i) d\mathbf{x}, \quad i = 1, ..., K$$

Many methods have been proposed in the literature so far trying to satisfy both the conditions at the same time and the most famous one is the iterative algorithm proposed by Linde, Buzo and Gray (LBG).

2.2.1 Case 1 - Source pdf $f_{\mathbf{X}}(\mathbf{x})$ known

1. Start from an initial codebook $\{\mathbf{y}_1^{(0)}, ..., \mathbf{y}_K^{(0)}\}$, set $n = 1$, $D^{(0)} = \infty$ (i. e. max distortion) and select a threshold $\epsilon > 0$

2. Compute the optimal partition exploiting the Nearest Neighbor Condition:

$$I_i^{(n)} = \{\mathbf{x} \in \mathbf{R}^L \text{ such that } ||\mathbf{x} - \mathbf{y}_i^{(n-1)}||_2^2 \leq ||\mathbf{x} - \mathbf{y}_j^{(n-1)}||_2^2, \ i \neq j\}, \ i = 1, ..., K$$

3. Compute the new codebook exploiting the Centroid Condition:

$$\mathbf{y}_i^{(n)} = \int_{I_i^{(n)}} \mathbf{x} f_{X|\mathbf{x}\in I_i^{(n)}}(\mathbf{x}|\mathbf{x} \in I_i^{(n)}) d\mathbf{x}, \quad i = 1, ..., K$$

4. Compute the total distortion:

$$D^{(n)} = \sum_{i=1}^{K} \int_{I_i} ||\mathbf{x} - \mathbf{y}_i^{(n)}||_2^2 f_X(\mathbf{x}) d\mathbf{x}$$

5. If $\frac{D^{(n-1)} - D^{(n)}}{D^{(n)}} < \epsilon$ then stop;

   else set n=n+1 and go to step 2.

Notice that this procedure does not guarantee to reach a global minimun of the distortion (however it does guarantee that the distortion from one iteration to the next one will not increase) and very much depends on the initial codebook. Notice also that this procedure needs to know the actual pdf $f_X(x)$ and computes two L-dimensional integers at each iteration at steps 3 and 4.

2.2.2 Case 2 - Source pdf $f_{\mathbf{X}}(\mathbf{x})$ unknown

If we don't know the pdf we can think of exploiting the empirical data building a training set $\mathcal{T} = \{\mathbf{x}_1, ...\mathbf{x}_N\}$ and then:

1. Given $\mathcal{T}$, start with an initial codebook $\{\mathbf{y}_1^{(0)}, ..., \mathbf{y}_K^{(0)}\}$, set $n = 1$, $D^{(0)} = \infty$ and $\epsilon > 0$.

2. Compute the optimal decision cells:

$$I_i^{(n)} = \{\mathbf{x} \in \mathcal{T} \text{ such that } ||\mathbf{x} - \mathbf{y}_i^{(n-1)}||_2^2 \leq ||\mathbf{x} - \mathbf{y}_j^{(n-1)}||_2^2,\ i \neq j\},\ i = 1, ..., K$$

3. Compute the new codebook:

$$\mathbf{y}_i^{(n)} = \frac{1}{|I_i^{(n)}|} \sum_{\mathbf{x} \in I_i^{(n)}} \mathbf{x}$$

   Warning: $I_i^{(n)}$ must be non-empty (if this happens then we change the representative of that decision cell with a vector that belongs to the cell that has the highest cardinality).

4. Compute the distortion

$$D^{(n)} = \frac{1}{|\mathcal{T}|} \sum_{\mathbf{x} \in \mathcal{T}} ||\mathbf{x} - Q(\mathbf{x})||_2^2,$$

   where obviously $|\mathcal{T}| = N$.

5. If $\frac{D^{(n-1)} - D^{(n)}}{D^{(n)}} < \epsilon$ then stop;

   else set n=n+1 and go to step 2.

Again notice that the codebook initialization is of fundamental importance for the performances of the entire algorithm, hence several techniques have been proposed such that random coding, pruning, splitting, Pairwise Nearest Neighbour, Product code and so on.
In this report I have always used the original methodology proposed by the authors: i. e. the **splitting technique** [2]. In this method we begin by designing a VQ with a single output point: with a codebook of just one element, the quantization region is the entire input space, hence the output point is the average value of the entire training set. From this point we can obtain a two-level VQ by including the output point for the one-level quantizer and a second output point obtained by adding a fixed perturbation vector (generally selected at random). Then we use the LBG algorithm to obtain the two-level vector quantizer until convergence and so on and so forth up to the point when the cardinality of the codebook is K.

Throughout the project I always used $\epsilon = 0.1$ and a fixed perturbation of 0.1.

2.3 Matlab Implementation

I wrote some Matlab scripts which are briefly presented in the following and are partially commented also inside the code:

- `umberto.m`: it is the main script to be ran. It loads the training signal and the signal we want to code using the LBG algorithm; starting from these it builds the L-dimensional sets. Then it performs the LBG algorithm and the resulting SNR are shown during the enlargement of the codebook.
- `loadAudio.m`: it just invokes the `audioread` function and, given the file name, it returns the signal x, the sampling frequency F_s and the number of bits per sample.
- `getVectors`: it is a routine used for the construction of the L-dimensional sets. In input is required the original double-channel signal, the desired vector dimension L and a string specifying whether we are coding the *mono* audio or the complete double-channel signal. Further remarks are discussed later.
- `LBG.m`: it is the function that computes the LBG algorithm. It computes the optimal codebook of the training set T. First of all it splits the codevectors and then it refines the solution exploiting the algorithm presented in section 2.2.2.
- `get_voronoi_regions.m`: it computes the Voronoi regions I_i, using the 2-norm given the set T and the codebook y. It basically computes the codevector at minimum l^2 distance from each of the L-dimensional vectors in T.
- `getSignal.m`: given a matrix of L-dimensional vectors it computes the reproducible audio signal.

2.4 Complications found

The debug of the code was a big effort also due to the computational demand required.

3 Results

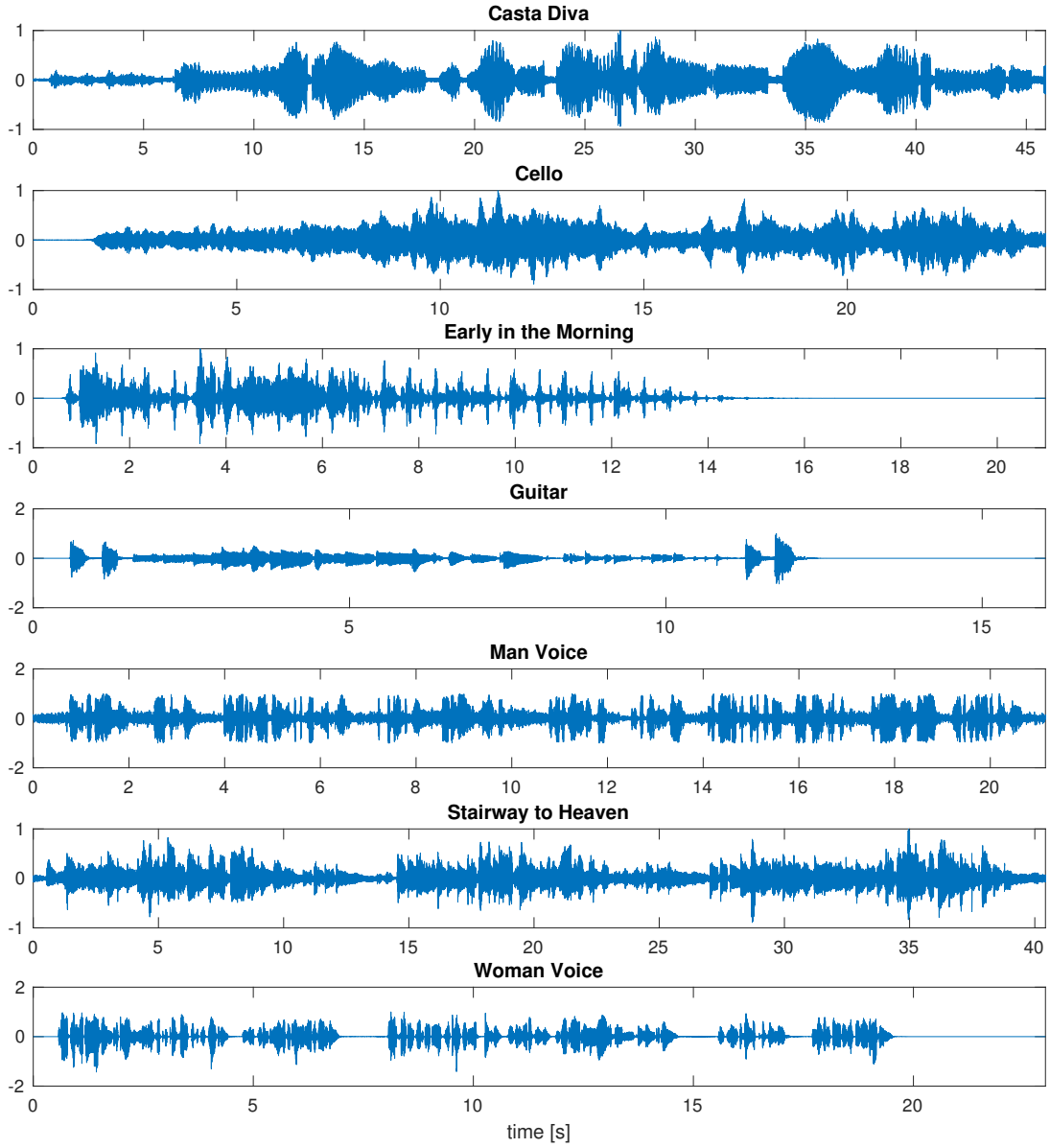

Figure 2: Database songs.

In this section are reported some results and all the of them are attached to this file inside the folder `Audio`. The dataset chosen is composed by a part, for lighter computations, of the following songs:

1. *Casta Diva:* it is an opera from the *Norma* by Vincenzo Bellini.

2. *Cello:* it is the *Cello Suite No.1 in G major* by Bach.

3. *Early in the mornin':* a country song by Eddie Rabbitt.

4. *Guitar:* it is a Spanish guitar-part of *Zapateado* by Larry Coryell.

5. *Man voice:* it is an excerpt from the famous speech by Charlie Chaplin in the *Great Dictator* (1940).

6. *Stairway to Heaven:* it is a part of the famous namesake song by Led Zeppelin (1971).

7. *Woman voice:* it is taken from the proposed database suggested by the professor.

The songs are plotted in Figure 2, just to have an idea.

3.1 Results for L=2, mono signals

In this section the results of compression using as vector dimension $L = 2$ and *mono* signals are presented fixing a maximum rate R of 4 bits per component. The resulting audio files can be found and played in the attached folder `Audio/lbg_compression/L2R4`.

3.1.1 Training audio is inside the dataset

In the considered scenario the *mono* signal is grouped into 2-dimensional vectors selected contiguous along the column. In this way we halve the length of the vector but we have two columns instead of just one.
Furthermore we can think of coding every audio files with one single training audio file (as done in practice because in a typical communication scenario the codebook needs to be sent to the receiver in order to properly decode the signal) or coding every audio files with the codebook computed on the audio itself. Intuitively we might expect a better result (in term of SNR) for the audio file coded using the codewords computed on the same signal; this because we are building the optimal codewords for the same signal.

Table 1 shows the results of coding the chosen dataset using as training set the audio *Casta Diva*. We can correctly notice that as K (the number of codevectors into the codebook) increase, then it increases also the SNR (expressed in dB in the Table); however this does require an higher rate.
Moreover we can notice that the SNR of *Casta Diva* is generally greater that the other retrieved signals, as discussed above. The only unusual data regards the *Cello* signal because sometimes its SNR is slightly greater that the SNR of *Casta Diva*; this can be due to the fact that *Cello* has an higher variance than *Casta Diva* (almost two times higher) and the samples are closer to the codewords in mean sense.

Also, it is interesting to see that the man voice is not well-captured by a woman singing an opera, as we can expect.

Audio name	*L*	*K*	*R bit/sample*	$\mathrm{SNR_{dB}}$	*K*	*R bit/sample*	$\mathrm{SNR_{dB}}$
Casta Diva	**2**	**2**	**0.5**	**2.8725**	**4**	**1**	**7.4876**
Cello	2	2	0.5	2.6925	4	1	7.3330
Early in the Morning	2	2	0.5	0.8704	4	1	5.6790
Guitar	2	2	0.5	-1.8154	4	1	4.3871
Man Voice	2	2	0.5	1.4508	4	1	3.2558
Stairway to Heaven	2	2	0.5	-1.9086	4	1	4.5466
Woman Voice	2	2	0.5	1.3290	4	1	6.1607
Casta Diva	**2**	**8**	**1.5**	**12.9501**	**16**	**2**	**17.8592**
Cello	2	8	1.5	12.5083	16	2	17.6690
Early in the Morning	2	8	1.5	10.0789	16	2	13.4206
Guitar	2	8	1.5	10.2144	16	2	15.5869
Man Voice	2	8	1.5	4.9777	16	2	6.1117
Stairway to Heaven	2	8	1.5	10.0850	16	2	15.3314
Woman Voice	2	8	1.5	10.3523	16	2	13.1654
Casta Diva	**2**	**32**	**2.5**	**21.9642**	**64**	**3**	**24.6008**
Cello	2	32	2.5	22.0293	64	3	24.9176
Early in the Morning	2	32	2.5	15.7615	64	3	17.2942
Guitar	2	32	2.5	19.3825	64	3	21.2062
Man Voice	2	32	2.5	6.9639	64	3	7.6450
Stairway to Heaven	2	32	2.5	19.2954	64	3	21.2993
Woman Voice	2	32	2.5	14.6991	64	3	15.4456
Casta Diva	**2**	**128**	**3.5**	**25.9558**	**256**	**4**	**29.6845**
Cello	2	128	3.5	26.5651	256	4	29.8972
Early in the Morning	2	128	3.5	18.6354	256	4	21.1628
Guitar	2	128	3.5	21.8688	256	4	23.9498
Man Voice	2	128	3.5	8.4725	256	4	9.4948
Stairway to Heaven	2	128	3.5	22.0431	256	4	25.8277
Woman Voice	2	128	3.5	15.9338	256	4	17.2472

Table 1: Training Set: *Casta Diva*

The curves of rate versus SNR using as training audio the *Casta Diva* signal are shown in Figure 3. We can match this figure with the data reported in Table 1 and we can confirm the previous considerations.

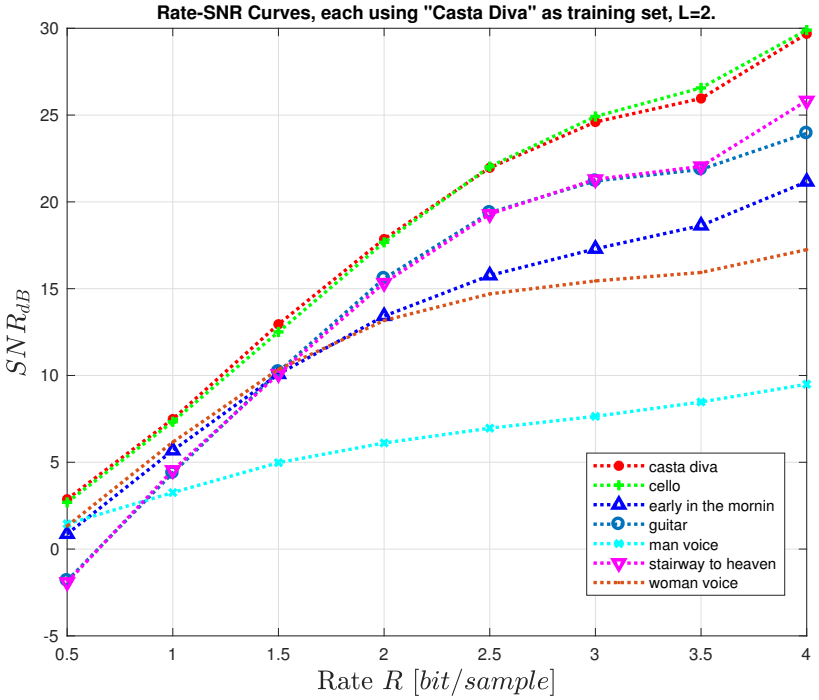

Figure 3: Rate-SNR curves using as training set *Casta Diva*, L=2.

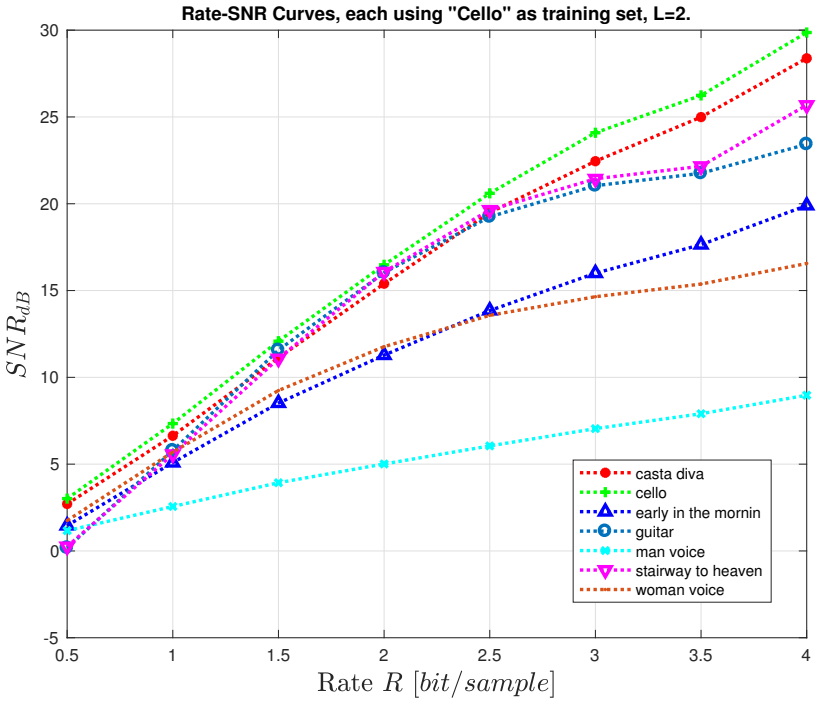

Figure 4: Rate-SNR curves using as training set *Cello*, L=2.

In order to verify the results I have also computed the SNRs using as training signal *Cello*, see Table 2 and the respective curves of rate versus SNR reported in Figure 3. We can notice that the respective values of the SNR for *Cello* are slightly higher in this case than before and it is the signal with highest SNR.

Audio name	*L*	*K*	*R bit/sample*	SNR_{dB}	*K*	*R bit/sample*	SNR_{dB}
Casta Diva	2	2	0.5	2.7077	4	1	6.6342
Cello	**2**	**2**	**0.5**	**3.2137**	**4**	**1**	**7.7372**
Early in the Morning	2	2	0.5	1.4590	4	1	5.0890
Guitar	2	2	0.5	0.1748	4	1	5.7820
Man Voice	2	2	0.5	1.1689	4	1	2.5618
Stairway to Heaven	2	2	0.5	0.2545	4	1	5.5669
Woman Voice	2	2	0.5	1.7495	4	1	5.6956
Casta Diva	2	8	1.5	11.1257	16	2	15.3945
Cello	**2**	**8**	**1.5**	**12.9817**	**16**	**2**	**17.9830**
Early in the Morning	2	8	1.5	8.5180	16	2	11.2798
Guitar	2	8	1.5	11.5700	16	2	16.0306
Man Voice	2	8	1.5	3.9304	16	2	5.0097
Stairway to Heaven	2	8	1.5	11.0800	16	2	16.0756
Woman Voice	2	8	1.5	9.2422	16	2	11.7660
Casta Diva	2	32	2.5	19.4903	64	3	22.4511
Cello	**2**	**32**	**2.5**	**22.6010**	**64**	**3**	**25.2899**
Early in the Morning	2	32	2.5	13.8465	64	3	15.9999
Guitar	2	32	2.5	19.2470	64	3	21.0405
Man Voice	2	32	2.5	6.0507	64	3	7.0455
Stairway to Heaven	2	32	2.5	19.6404	64	3	21.4372
Woman Voice	2	32	2.5	13.5712	64	3	14.6455
Casta Diva	2	128	3.5	24.9918	256	4	28.3792
Cello	**2**	**128**	**3.5**	**27.2407**	**256**	**4**	**30.3643**
Early in the Morning	2	128	3.5	17.6436	256	4	19.8987
Guitar	2	128	3.5	21.7530	256	4	23.4265
Man Voice	2	128	3.5	7.9044	256	4	8.9703
Stairway to Heaven	2	128	3.5	22.1617	256	4	25.6822
Woman Voice	2	128	3.5	15.3736	256	4	16.5595

Table 2: Training Set: *Cello*

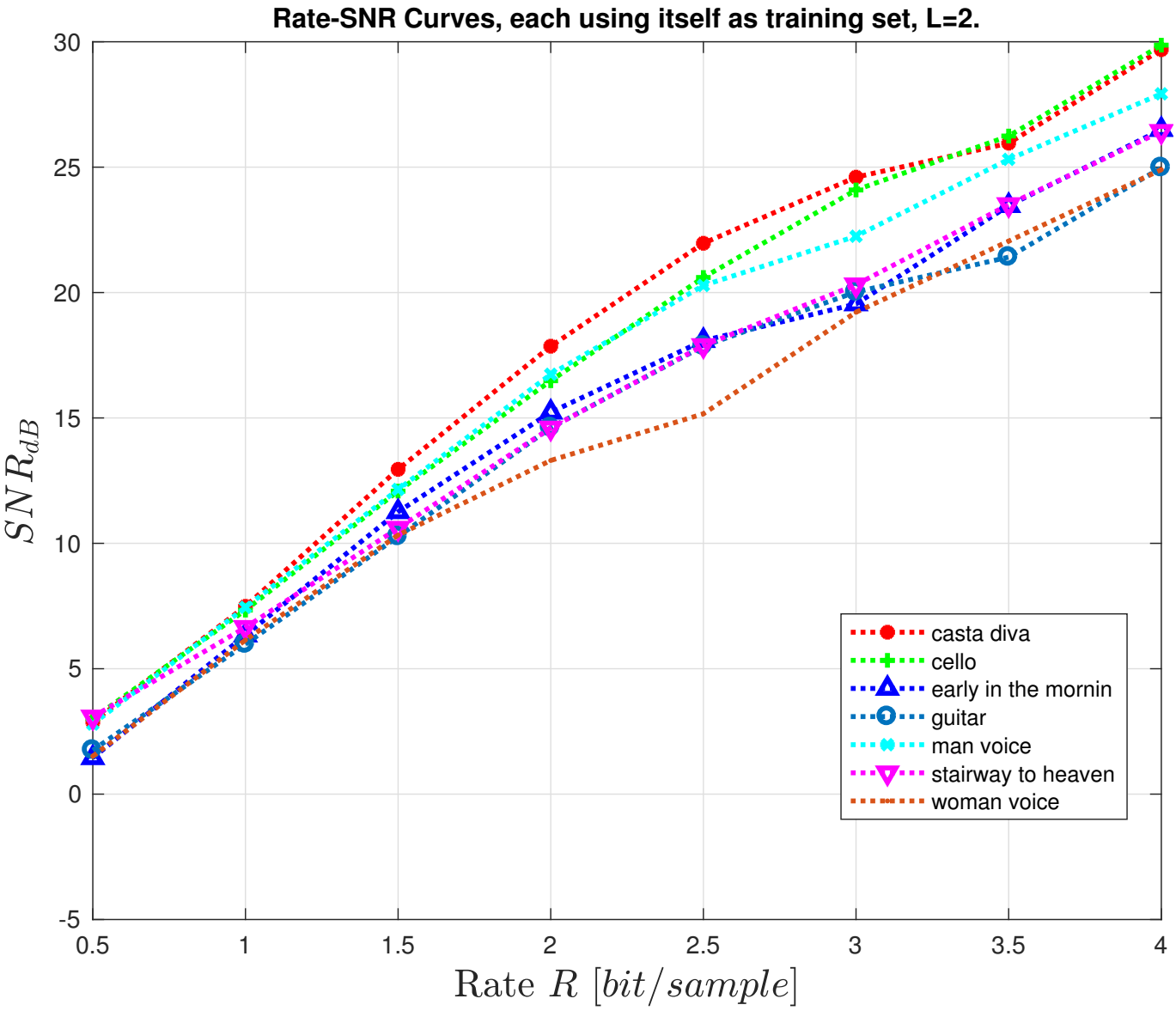

Figure 5: Rate-SNR curves using itself as training set, L=2.

Just to clarify, I computed the values of SNR of each signal using itself as training audio. The resulting plot is shown in Figure 5, where we can notice that every SNR-curve has been pulled up thanks to the choice of the training set. On the other side, however, the codebook needs to be retransmitted every time. Compare this plot with the curves shown in Figures 3 and 4.

3.1.2 Training audio is mixed

Another aspect which can be evaluated is the usage of a mixed training audio.

I distinguished two possible mixed training dataset:

1. using pieces of audio of the signals to send;

2. using various pieces of audio (songs, speeches,...).

Using the first proposal I observed the results shown in Table 3. You can compare these results with the Tables 1 and 2; in particular you can see that in general the value of the SNR lays in between the SNR achievable using the same signal as training set or another one.

Audio name	***L***	***K***	***R bit/sample***	$\mathbf{SNR_{dB}}$	***K***	***R bit/sample***	$\mathbf{SNR_{dB}}$
Casta Diva	2	2	0.5	2.8727	4	1	6.8856
Cello	2	2	0.5	2.8653	4	1	7.5375
Casta Diva	2	8	1.5	11.3703	16	2	16.0073
Cello	2	8	1.5	12.9574	16	2	17.8324
Casta Diva	2	32	2.5	21.9401	64	3	24.4505
Cello	2	32	2.5	22.1432	64	3	24.9786
Casta Diva	2	128	3.5	25.8274	256	4	28.9885
Cello	2	128	3.5	26.8690	256	4	29.9532

Table 3: Training Set: *Mixed 1*, mono audio.

Using the second proposal I obtained the results shown in Table 4, similar results to what observed above.

Audio name	***L***	***K***	***R bit/sample***	$\mathbf{SNR_{dB}}$	***K***	***R bit/sample***	$\mathbf{SNR_{dB}}$
Casta Diva	2	2	0.5	2.8169	4	1	7.6483
Cello	2	2	0.5	2.7231	4	1	6.9820
Casta Diva	2	8	1.5	12.8583	16	2	16.8728
Cello	2	8	1.5	11.4365	16	2	17.4632
Casta Diva	2	32	2.5	21.8694	64	3	24.4505
Cello	2	32	2.5	22.3214	64	3	24.6318
Casta Diva	2	128	3.5	25.7687	256	4	29.0794
Cello	2	128	3.5	27.0243	256	4	29.8523

Table 4: Training Set: *Mixed 2*, mono audio.

Therefore, we can conclude that the distortion introduced by the LBG quantization algorithm does depend mostly on the input audio and not on the selection of the training set; of course extreme cases are possible.

3.1.3 Varying ϵ value

Up to this point we have always considered a fixed value for ϵ and in particular it was set to 0.1; in this section we are going to see what does it change in terms of SNR by varying this parameter. The result is shown in Table 5, where we can confirm the common sense. In fact as ϵ decreases the SNR increases, because we are performing more iterations of the algorithm in order to be closer to the codevectors.

Audio name	*L*	ϵ	*K*	*R bit/sample*	SNR_{dB}	ϵ	*K*	*R bit/sample*	SNR_{dB}
Casta Diva	2	0.001	2	0.5	3.0152	0.005	2	0.5	2.8725
Casta Diva	2	0.001	4	1	7.9870	0.005	4	1	7.8220
Casta Diva	2	0.001	8	1.5	13.5375	0.005	8	1.5	13.4462
Casta Diva	2	0.001	16	2	18.6041	0.005	16	2	18.5171
Casta Diva	2	0.001	32	2.5	22.7208	0.005	32	2.5	22.5921
Casta Diva	2	0.001	64	3	25.2944	0.005	64	3	25.0320
Casta Diva	2	0.001	128	3.5	27.9557	0.005	128	3.5	27.8786
Casta Diva	2	0.001	256	4	30.6781	0.005	256	4	30.6668
Casta Diva	2	0.01	2	0.5	2.8725	0.05	2	0.5	2.8725
Casta Diva	2	0.01	4	1	7.7854	0.05	4	1	7.4876
Casta Diva	2	0.01	8	1.5	13.3882	0.05	8	1.5	12.9501
Casta Diva	2	0.01	16	2	18.4178	0.05	16	2	17.8592
Casta Diva	2	0.01	32	2.5	22.5019	0.05	32	2.5	21.9642
Casta Diva	2	0.01	64	3	24.9349	0.05	64	3	24.6008
Casta Diva	2	0.01	128	3.5	27.7758	0.05	128	3.5	27.3036
Casta Diva	2	0.01	256	4	30.5321	0.05	256	4	30.1607
Casta Diva	2	0.1	2	0.5	2.8725	0.2	2	0.5	2.8725
Casta Diva	2	0.1	4	1	7.4876	0.2	4	1	7.0128
Casta Diva	2	0.1	8	1.5	12.9501	0.2	8	1.5	11.8937
Casta Diva	2	0.1	16	2	17.8592	0.2	16	2	16.9363
Casta Diva	2	0.1	32	2.5	21.9642	0.2	32	2.5	21.2211
Casta Diva	2	0.1	64	3	24.6008	0.2	64	3	24.1435
Casta Diva	2	0.1	128	3.5	25.9558	0.2	128	3.5	25.7855
Casta Diva	2	0.1	256	4	29.6845	0.2	256	4	29.0137

Table 5: Training Set: *Casta Diva*, mono channel. Effects of varying ϵ.

Furthermore it is worth to notice that if we select, for example, $\epsilon = 0.001$ then the audio signal *Cello* coded using as training audio the signal *Casta Diva* has an SNR higher than the *Casta Diva*'s one, as reported in Table 6 (compare with Table 2).

Audio name	*L*	ϵ	*K*	*R bit/sample*	SNR_{dB}	ϵ	*K*	*R bit/sample*	SNR_{dB}
Casta Diva	2	0.001	2	0.5	3.0152	0.001	4	1	7.9870
Casta Diva	2	0.001	8	1.5	13.5375	0.001	16	2	18.6041
Casta Diva	2	0.001	32	2.5	22.7208	0.001	64	3	25.2944
Casta Diva	2	0.001	128	3.5	27.9557	0.001	256	4	30.6781
Cello	2	0.001	2	0.5	2.6925	0.001	4	1	7.2349
Cello	2	0.001	8	1.5	12.5054	0.001	16	2	17.8662
Cello	2	0.001	32	2.5	22.3379	0.001	64	3	25.3148
Cello	2	0.001	128	3.5	27.5530	0.001	256	4	30.2206

Table 6: Training Set: *Casta Diva*, mono. channel. Effects of varying ϵ

Audio name	*L*	*K*	*R bit/sample*	SNR_{dB}	*K*	*R bit/sample*	SNR_{dB}
Casta Diva	**4**	**2**	**0.25**	**2.8399**	**4**	**0.5**	**7.3190**
Cello	4	2	0.25	2.6684	4	0.5	7.2046
Early in the Morning	4	2	0.25	0.8327	4	0.5	5.4771
Guitar	4	2	0.25	-1.8102	4	0.5	4.2824
Man Voice	4	2	0.25	1.4270	4	0.5	3.1842
Stairway to Heaven	4	2	0.25	-1.9352	4	0.5	4.3565
Woman Voice	4	2	0.25	1.0949	4	0.5	5.5637
Casta Diva	**4**	**8**	**0.75**	**11.9330**	**16**	**1**	**15.8205**
Cello	4	8	0.75	11.8217	16	1	16.1003
Early in the Morning	4	8	0.75	9.1571	16	1	11.8981
Guitar	4	8	0.75	9.7466	16	1	13.8769
Man Voice	4	8	0.75	4.6066	16	1	5.7378
Stairway to Heaven	4	8	0.75	9.2433	16	1	13.1278
Woman Voice	4	8	0.75	8.6132	16	1	10.3795
Casta Diva	**4**	**32**	**1.25**	**18.1372**	**64**	**1.5**	**20.3176**
Cello	4	32	1.25	18.7415	64	1.5	20.6696
Early in the Morning	4	32	1.25	13.5667	64	1.5	15.1475
Guitar	4	32	1.25	15.8934	64	1.5	16.4440
Man Voice	4	32	1.25	6.5727	64	1.5	7.4134
Stairway to Heaven	4	32	1.25	14.9810	64	1.5	16.3370
Woman Voice	4	32	1.25	11.1671	64	1.5	11.9884
Casta Diva	**4**	**128**	**1.75**	**23.3185**	**256**	**2**	**26.2502**
Cello	4	128	1.75	23.4501	256	2	26.2850
Early in the Morning	4	128	1.75	16.9731	256	2	18.4212
Guitar	4	128	1.75	18.6718	256	2	20.2897
Man Voice	4	128	1.75	8.2594	256	2	9.2166
Stairway to Heaven	4	128	1.75	19.2748	256	2	21.6565
Woman Voice	4	128	1.75	12.9532	256	2	13.9245

Table 7: Training Set: *Casta Diva*

3.2 Results for L=4, mono signals

In this section is discussed the situation symmetric of the above one; i. e. here we select $L = 4$ and a maximum rate of $R = 2$ bits per component.

The result is shown in Table 7, where we used *Casta Diva* as training set. The table has to be compared with Table 1; we can notice that, at the same values of K and complexity, the SNRs are always lower in this case because we are increasing the quantization noise since the codewords have higher dimension than before. However in terms of rate this situation is preferable. Hence it is a trade-off and it depends on the targeted application.

Reproducing the resulting audio signals one can effectively hear that, at the same values of K, the signal coded using itself as training set is the *best* one, i. e. it is composed with the right tonalities and it is the

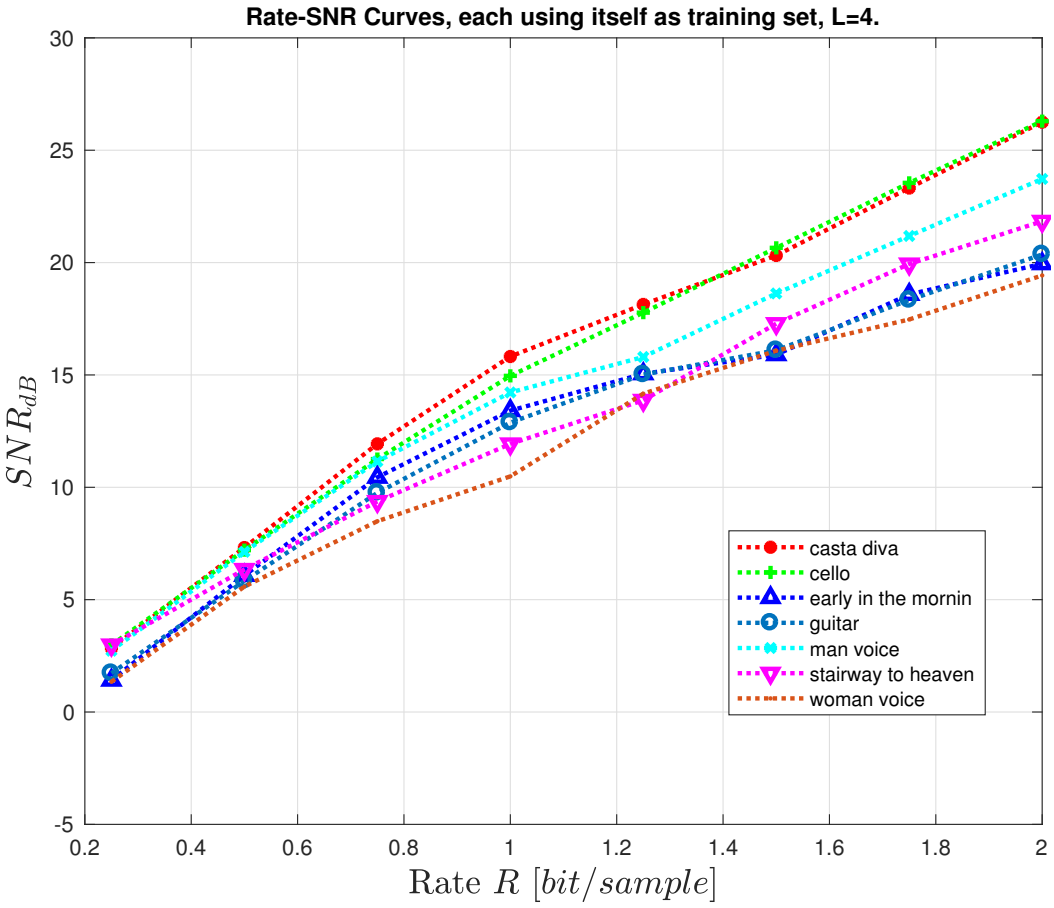

Figure 6: Rate-SNR curves using itself as training set, L=4.

easiest one to be recognized. On the signals coded using a different training set we can hear the artifacts and sometimes it is possible to match them with the tonalities of the training signal.
A quick demonstration will be given during the oral presentation.

Furthermore in Figure 6 we can see the rate-SNR curves using $L = 4$ and the signal itself as training audio; we can compare this plot with Figure 5 and we can confirm the considerations above of lower rates but also lower SNRs.

Unfortunately the case study with $L = 4$ and $R = 4$ was too computationally complex inducing a number of $K = 2^{16} = 65536$ codewords in the codebook.

3.3 Results for L=2, double-channel signals

Another method of encoding audio signals could be trying to exploit the correlation among the two channel of the signal. Fixing L=2 and varying the rate up to 4 bit/sample we can build similar tables and graphs as the ones reported above.

For example we can look at the situation in which we code the signals using *Casta Diva* as training set. In Table 8 we can appreciate that the values are higher than before (see Table 1 as comparison). The rate is

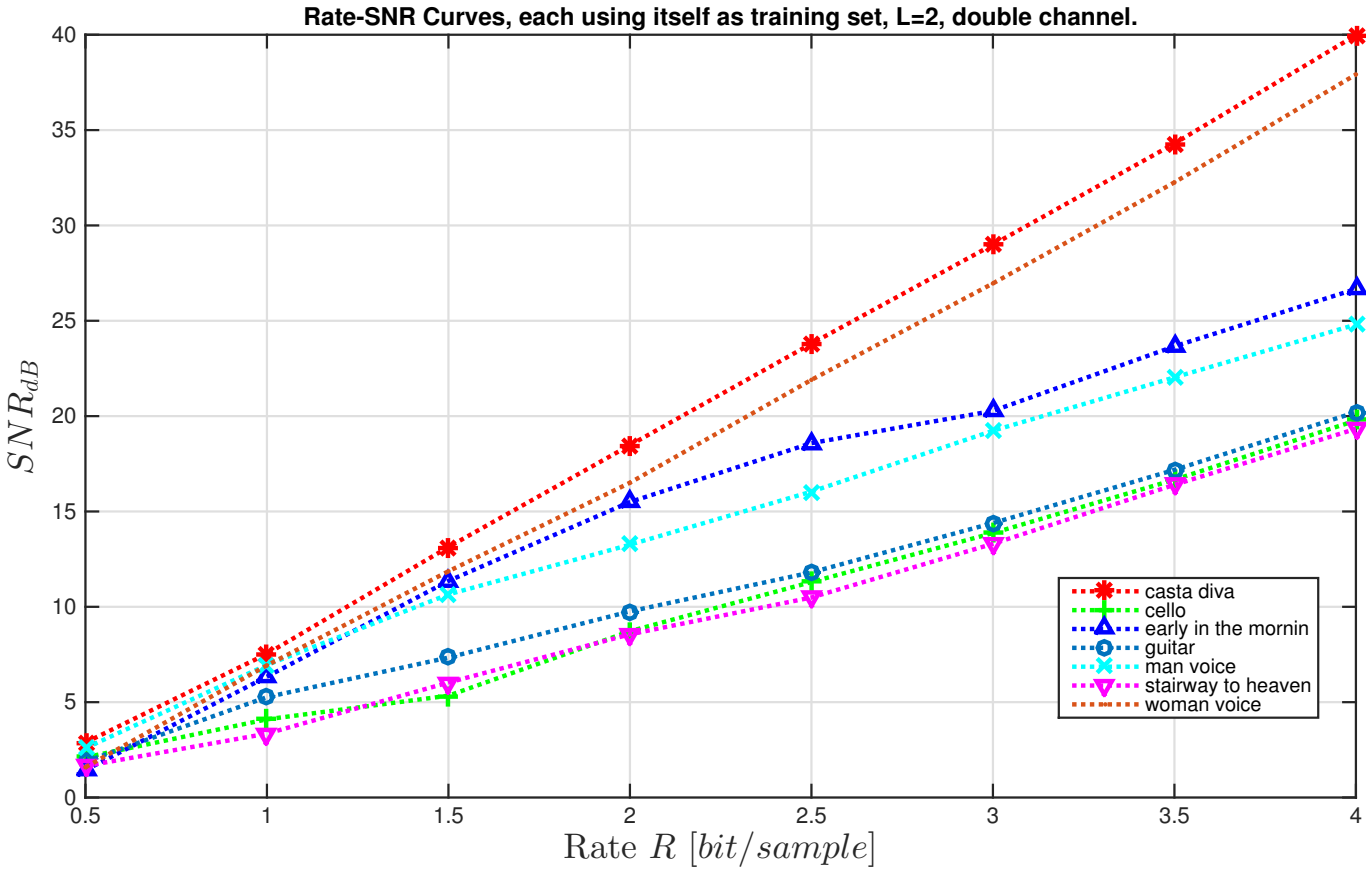

Figure 7: Rate-SNR curves using itself as training set, L=2, double channel.

now doubled because we are transmitting both the channels.

Audio name	***L***	***K***	***R bit/sample***	$\mathbf{SNR_{dB}}$	***K***	***R bit/sample***	$\mathbf{SNR_{dB}}$
Casta Diva	2	2	1	2.8811	4	2	7.5323
Casta Diva	2	8	3	13.1352	16	4	18.4770
Casta Diva	2	32	5	23.7996	64	6	28.9906
Casta Diva	2	128	7	34.2952	256	8	39.9759

Table 8: Training Set: *Casta Diva*, double channel.

Otherwise we can look at the rate-SNR curves obtained coding with training set the signal itself. The Figure 7 illustrates the behavior; comparing it with the Figure 5 we can notice that the majority of the values are higher now than before as we expected. However, some of them like cello, guitar, stairway to heaven are not improved because the correlation between the two channels is low due to noise or other things (this can be also seen empirically by plotting the two channels and comparing them, which it turns out that in some cases are *really* different).

4 Conclusions

In this report we have seen an application of the LBG algorithm to audio signals analyzing some performances varying many parameters' settings.

We can hear really good results and we can testify them in terms of the SNR, nevertheless it is worth to notice that the LBG is too much computationally demanding and it requires to transmit the codebook to the receiver each time it is changed. Hence in practice it is not used in favor of some others well-known coding procedures for audio signals such as MP3, MPEG-4 or Vorbis.

References

[1] K. Sayood, *Introduction to data compression*. Newnes, 2012.

[2] Y. Linde, A. Buzo, and R. Gray, "An algorithm for vector quantizer design," *IEEE Transactions on communications*, vol. 28, no. 1, pp. 84–95, 1980.