Project # 3. Multimedia Coding

Alexandre Rodrigues (2039952)

January 12, 2022

1 Introduction

This reports is dedicated to explain the usage of the LBG-split algorithm and how to implement it. The Linde-Buzo-Gray algorithm is a lossy coding technique, it uses vector quantization, meaning that a block of input samples (size L) is processed together. ...

2 Technical Approach

Each vector of size L samples can be defined as

$$x = [x_1, x_2, \dots, x_L], x \in R^L \tag{1}$$

where $x_i, i = 1, 2, \dots, L$ are input samples. Using y_i as a codevector,

$$B = \{y_1, y_2, \dots, y_L\} \tag{2}$$

is the set of reconstruction levels, i.e. the codebook, of size K. The decision cells will be

$$I_i \in R^L, i = 1, 2, \dots, K, such that I_i intersect I_j = 0 \aleph i \neq j and union k i = 1 I_i = R^L \tag{3}$$

2.1 LBG

Since we do not know the probablity distributuion function of the input data $f_x(x)$ we can use a training set

$$T = x_1, \dots, x_N \tag{4}$$

, where N should be considerably larger than the codebook size K,$N \leq 500K$ for this work.

The algorithm can then be defined as:

1. initial codebook
2. optimal partition
3. new codebook
4. distortion
5. terminate?

2.2 Split approach

This approach is based on starting a codebook as

$$\{(1-\epsilon)y_{avg}, (1+\epsilon)y_{avg}\}, \tag{5}$$

where y_{avg} is the average of the vectors in the training set. The LBG algorithm is then applied to this codebook. The returning optimized codebook is split in the same way, i.e.

$$\{(1-\epsilon)y_i, (1+\epsilon)y_i, \ldots, (1-\epsilon)y_N, (1+\epsilon)y_N\}, \tag{6}$$

This implies that the codebook size will double in each iteration until we get the desired size K, $N = 2, 4, 8, \ldots, K$.

3 Results

There were several tests made to fully understand the performance of this method.

3.1 Important Parameters

As required there are 4 scenrarios

L	**R**	**K**
2	2	16
2	4	256
4	1	16
4	2	256

Table 1: Scenarios

where K is the codebook size

$$K = 2^{RL}. \tag{7}$$

3.2 Training Sets

There 3 different trainng sets used:

- All audio files from MC dataset
- Only the Say Nada song
- All Audio Files and All Popular Music

Each training set was limited in size. For each file I extracted the middle part to result in the following relative size (N/K).

Training set	**N/K for L = 2**	**N/K for L = 4**
All Audio	1250	625
Say Nada	1000	500
All Music and Audio	4844	2422

Table 2: Relative Sizes of each Training Set

These sizes allowed fast enough codebook computation. Having training sets with and without including the encoding objects will benefit our comparison and possible conclusions.

3.3 Training Performance

The tests were made using $\epsilon = 0.01$.

Training set	**2,2**	**2,4**	**4,1**	**4,2**
All Audio	1.51×10^5	3.53×10^5	4.48×10^5	3.15×10^6
Music: Say Nada	4.36×10^6	3.71×10^6	3.31×10^7	2.70×10^7
All Music and Audio	2.47×10^6	1.93×10^6	9.70×10^6	1.42×10^7

Table 3: Distortion for each training set and each values of L and R

Training set	**2,2**	**2,4**	**4,1**	**4,2**
All Audio	$1.29s$	$179.09s$	$0.45s$	$86.025s$
Music: Say Nada	$0.75s$	$114.41s$	$0.47s$	$55.81s$
All Music and All Audio	$2.57s$	$556.13s$	$1.21s$	$266.74s$

Table 4: Time for each training set and each values of L and R

We can see that (4,1) is clearly the fastest training. The training time is mostly dependent on the value K. The distortion is noticeably larger for L = 4.

3.4 Encoding Performance

In summary I got the following results:

Encoded	**2,2**	**2,4**	**4,1**	**4,2**
70mono	$1.50s$	$17.81s$	$0.74s$	$10.24s$
Average Music	$13.44s$	$188.31s$	$6.78s$	$95.41s$
Worst Case	$18.02s$	$137.08s$	$8.24s$	$118.51s$
Best Case	$9.00s$	$253.91s$	$4.63s$	$70.26s$

Table 5: Time for each encoding object and each values of L and R

There is no noticeable difference between using the different training sets.

Encoded	**2,2**	**2,4**	**4,1**	**4,2**
70mono	2.94×10^5	2.81×10^4	6.32×10^5	7.51×10^4
Average Music	3.40×10^6	3.35×10^5	4.38×10^6	7.30×10^5
Worst Case	1.53×10^7	1.95×10^6	1.51×10^7	2.85×10^6
Best Case	3.66×10^5	2.21×10^4	6.90×10^5	1.99×10^5

Table 6: Distortion for each encoding object and each values of L and R

Encoding music files using All Audio as training se gives us in average 3 times the distortion when using Say Nada as the training set. When using Audio and Music as the training set we got half the distortion compared to using Say Nada. For (2,4) we get actually 11 times the distortion.

Encoded	**2,2**	**2,4**	**4,1**	**4,2**
70mono	12.55		10.10	
Average Music	13.83		11.80	
Worst Case	17.01		16.60	
Best Case	8.74		8.00	

Table 7: SNR(dB) for each encoding object and each values of L and R

Regarding Signal to Noise Ratio (SNR) in dB, there are significant differences regarding the training set used.

Encoded	**2,2**	**2,4**	**4,1**	**4,2**
All Audio	10.35	20.31	9.66	17.25
Music: Say Nada	14.15	26.63	11.57	20.74
All Music and All Audio	16.98	27.06	14.17	21.05

Table 8: Average SNR(dB) for each training set and each values of L and R

3.5 Training set including the encoding object...

4 Conclusions

1 - Using music as a training set is clearly superior when we will use the codebook to encode music files. Vice-versa is also valid, although less significantly. Encoding 70mono using All audio as training set produces in average half the distortion.

2 -

3 -

4 -