CAV - Codificação Audio e Video
Guião 2

Francisco Oliveira (80108), Hugo Sampaio (79961) and Ricardo Chaves (79946)

***Abstract*—The main topic of this guide is audio coding. Lossless and lossy encoding techniques were explored. The codecs and auxiliary software were developed using Python 3.**

This report will discuss our process for lossless coding, lossy coding and finishes with some results regarding performance.

NOTE: In order to write individual bits and read bits, the BitArray class from the bitstring package was used, as it has a simple interface and automatic padding.

I. Golomb Coding

Golomb Coding is a lossless data compression method ideal for situations in which the occurrence of small values is more likely than large values. This is the type of coding used in this guide.

A value coded in Golomb is represented in two parts: a variable length unary part, and a fixed length binary part.

II. Lossless Coding

The goal is to have an encoded file, with information, that can be decoded to retrieve the same information as the original file. This process has no information loss, despite the fact that the encoded file is smaller in size.

To do so, we take advantage of temporal redundancy and variable length codes.

After reading the samples inside the audio file to be encoded, this information passes through a few stages, before it can be converted into the encoded file. These stages will be detailed in the following subsections.

A. Applying a Predictor

When reading the file and extracting the samples, a predictor is used. This predictor simply predicts that the following sample will have the same value as the previous one.

This allows the data set of the sample values to be transformed into another set of data, the allows us to take advantage of temporal redundancy. Instead of having a sequence of sample values, at the end of this stage, we have a sequence of values that represent the differences between the predicted value (last value) and the value the actually occurred at that temporal instance.

B. Statistical analysis

At this stage, it is computed the probability of each value in our data set. This data is considered after the predictor is applied and for each channel separately.

These probabilities are used to map these values that need to be encoded to our symbols.

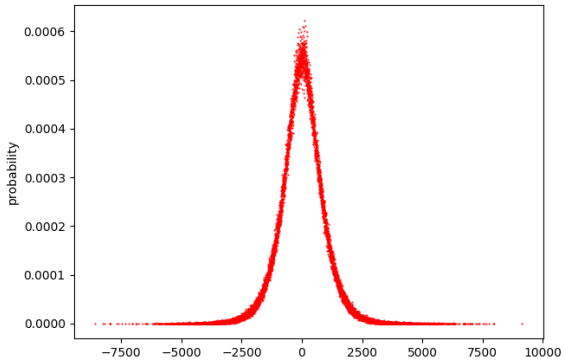

Fig. 1. Probability distribution of the values to be encoded after predictor is applied. (left channel sample01.wav)

C. Mapping values and symbols

After calculating the probability of the values to encode, each of them is mapped to a symbol. These symbols are represented by a non-negative integer values.

These symbols will be encoded in the following stage with a variable length code, so this mapping should be efficient. Therefore, most likely values should be mapped to a smaller symbol, as these will be coded with smaller code words, and less likely values should be mapped to a larger symbol, as these will be coded to larger codewords.

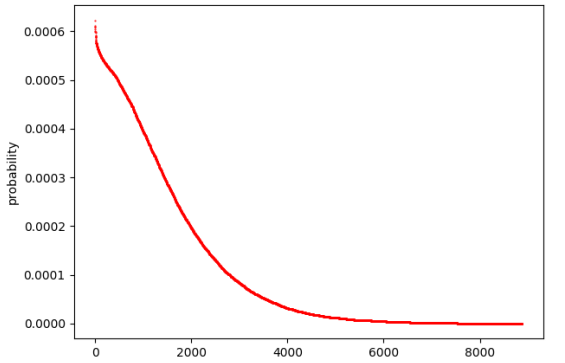

Fig. 2. Probability distribution of the symbols (left channel sample01.wav)

This mapping will also be used, when writing the encoded file, so that this mapping between symbols and values can be reverted for each channel.

D. Finding the best Golomb parameters

The symbols will be encoded using Golomb. The performance of this code is very dependent on the M parameter being adjusted to the data set being encoded.

Therefore, we try to find the best parameter for the symbols on each channel. This is done by fitting the probability values of each symbol to the ideal distribution curve for as information source. This curve is given by the equation 2 for Golomb encoding.

$$P(N) = \alpha^N \times (1 - \alpha) \tag{1}$$

where,

$$\alpha = ceil(-\frac{1}{log2(\alpha)}) \tag{2}$$

After fitting the data to this ideal function, the result is a curve from which the parameter is extracted. Here, on Figure 3, we have the data and the fitted curve overlapped.

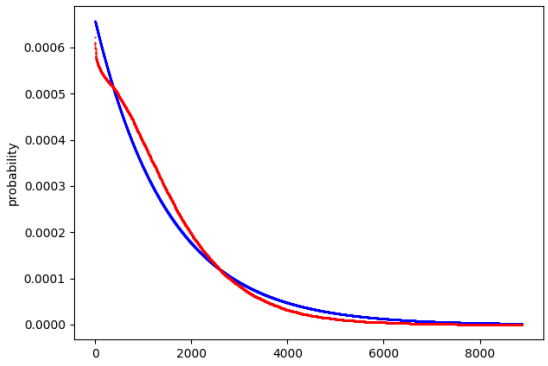

Fig. 3. Curve fit of the data set comprised of the symbols for the left channel of sample01.wav

E. Writing encoded file

Finally, the encoded file is generated.

After these steps the sequence of values to be encoded after the predictor is applied is mapped to symbol and symbols are encoded in Golomb with the found parameter. This process is done for each channel, so each channel may have a different set of symbols and may be encoded with different Golomb parameters.

This data is placed in a WAV formated file along side with a custom header, so that the file can later be decoded.

This header contains the parameters used for the Golomb encoding of each channel and also the symbol to value mapping for each channel. It is generated using the pickle.dumps() function of the Pickle serialization library in Python.

III. Lossy Coding

With lossy coding we quantize the prediction error, in this case the difference between the the actual value and the prediction value. With this process the prediction value will not be simply the previous value, but actually the sum of the reconstructed value of the prediction error and the previous value. When decoding we need to dequantize the value first and only after that can we calculate the actual value by adding the dequantized value with the previous value.

A. Quantization

For the quantization we used uniform quantization with the quantized value being calculated with:

$$Q(N) = floor(\frac{N}{2^B}) \tag{3}$$

with B being the difference between the original number of bits and the new number of bits (to represent a sample).

To dequantize we do the reverse operation:

$$Q(N) = N \times 2^B \tag{4}$$

IV. Decoding Encoded Files

When files are encoded, they are written with an added header block in the start of the file, after the WAVE header and before the data part. The decoding process starts by loading this data structure to memory, getting the maps that translate symbols back to value from the file, as well as the M values to be used in the Golomb decoding.

Each sample is then read, unpacked, translated from symbol to a value and added to the previous sample (remember that predictive coding is used).

V. Performance

The performance of the codecs can be rated using several ways. In Table I and Table II, we show the processing times (both for encoding and decoding), the compression ratios (originalSize/compressedSize) and the m value used for the Golomb encoding.

In Table III we show the processing times, the compression ratios and signal to noise ratio (SNR) for different values of quantization. Original size for a sample is 16 bits, that's why only values smaller than 16 are used.

file	encoding time (min)	decoding time (min)	m left channel	m right channel	compression ratio
sample01.wav	1:56	0:47	972	1055	1.317
sample02.wav	0:58	0:22	635	276	1.269
sample03.wav	1:18	0:30	353	349	1.402
sample04.wav	0:52	0:20	865	912	1.299
sample05.wav	1:21	0:33	175	137	1.442

TABLE I
Compression using the Lossless Codec

file	encoding time (min)	decoding time (min)	m left channel	m right channel	compression ratio	SNR
sample01.wav	2:01	0:44	4	4	4.333	28.78
sample02.wav	0:59	0:23	3	2	4.690	24.62
sample03.wav	1:21	0:30	2	2	5.490	25.21
sample04.wav	0:54	0:21	4	4	4.250	31.40
sample05.wav	1:23	0:30	2	2	6.150	26.61

TABLE II
Compression using the Lossy Codec

Sample size (in bits)	SNR
10	43.22
8	31.40
6	19.70
4	8.33

TABLE III
COMPRESSION OF SAMPLE04.WAV USING DIFFERENT VALUES FOR THE SAMPLE SIZE