Introduction to Lossy Compression

A raw audio or image file is not an exact representation of a physical quantity it is measuring. There are a few reasons for this. First, real physical sensors are noisy. If you have a cheap camera or microphone, then you cannot expect to get an error free measurement. Second, when the encoded image is displayed to a person (or when the sound is played from a speaker or earphone) there is always some noise present in the display. For images, a monitor or television screen (or a printer) is a physical device and it introduces noise. Same for audio. Plus, there are other signals present that add noise. e.g. If you listen to sound from a speaker, there is always some other sound in the room.

Even if we had an exact representation of a sound or image, it is unclear that we need it. When you listen to a sound or you look at an image or video, you are relatively insensitive to small differences in intensities. e.g. In an image, if you change the intensity of some pixel from 102 to 103, then a person looking at the image is unlikely to notice.

If we are going to allow "loss" of information (that is, errors) then how can we do so? How can one best compress signals in a *lossy* way, i.e. if the decoder only gets back an approximation of the original signal rather than the exact original signal ? There are a number of principles that need to be followed and we will learn about them over the next few lectures. We begin by looking at what we mean by an approximation.

Quantization

Suppose we have a *real valued random variable* X. This variable might take values from an infinite alphabet (the real line), or some very large finite alphabet (single precision floats, 32 bit integers, or even 16 bit integers). We wish to encode samples of this random variable by using one of N codewords, where N is smaller than the size of the alphabet of X. Here we use N in the same way as we did in lossless compression, namely, the number of symbols in an alphabet. However, this alphabet is smaller than the original alphabet: it is defined by subsets of the original alphabet.

Let's take the case that X is a real number. To define which codeword is assigned to each value of X, we partition the real line into N intervals with $N+1$ *boundaries* $B_0, B_1, B_2, \ldots, B_N$, where $B_0 = -\infty, \quad B_N = \infty$.

Let x be a particular *sample* of the random variable X. We define a *quantization function* $Q(X)$ such that the $Q(x)$ is the *representative value* of x, that is, $Q(x)$ serves as the approximation of x. The function $Q(X)$ has the reasonable property that if $B_i < x < B_{i+1}$, then $B_i < Q(x) \leq B_{i+1}$, i.e. it maps the interval $(B_i, B_{i+1}]$ to a particular value in that interval.

There are many ways to choose the boundaries B_i and the representative values. We concentrate on just one: the uniform quantizer.

Uniform Quantizer

For the uniform quantizer, the decision boundaries are uniformly spaced, that is,

$$B_i - B_{i-1} = \Delta, \text{ for } i = 2, 3, \ldots, N-1.$$

We define the *representative values* :

$$Q(x) = \begin{cases} B_1 - \frac{\Delta}{2}, & \text{if } x \leq B_1 \\ \\ B_i + \frac{\Delta}{2}, & B_i < x \leq B_{i+1} \text{ where } i \in \{1, ..., N-1\} \end{cases}$$

The idea is that the encoder sends the codeword for the interval number $i \in 1, \ldots, N$ and the decoder estimates x to be the representative value for the i^{th} interval, namely $Q(x)$. The encoder and decoder need to agree beforehand on what is N and what the N representative values $Q()$ are. This information would either be part of a standard, or part of a header file.

Two types of Quantization Error

The quantization error is $Q(x) - x$. This error comes in two forms:

1. *granular error:* if $x \in (B_1, B_{N-1}]$ then

$$| \; Q(x) - x \; | \; < \frac{\Delta}{2}$$

2. *overload error:* This is the case that $x > B_{N-1}$ or $x < B_1$. The error is potentially unbounded here.

Using a small Δ decreases the probability of having a granular error, but increases the probability of overload errors, and vice-versa. A small Δ also means that the granular errors that do occur will tend to be small.

Example

Let $B_1 = 5$ and $B_{N-1} = 35$ and $\Delta = 2$. How big is N ?

$$B_{N-1} - B_1 = (N-2)\Delta$$

Answer: $N = 17$. That is, 15 intervals where granular errors can occur, and two intervals where overflow error can occur.

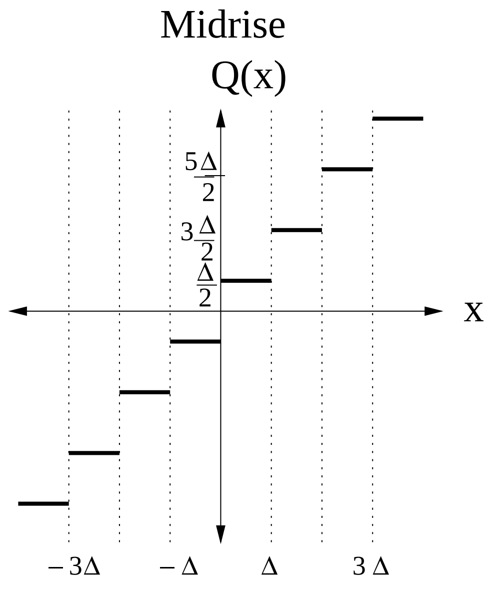
Mid-rise vs. mid-tread

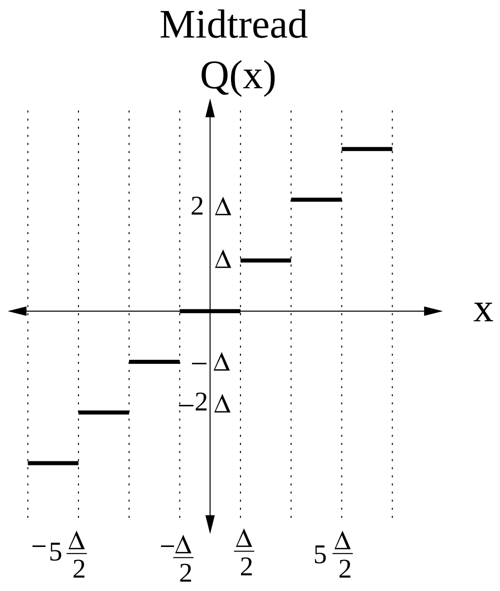
A very common case to consider is that the probability density of X is approximately symmetric about the value 0. Usually one defines one of the two following quantization functions in this case. These depend by whether N is even or odd.

- *mid-rise quantizer*: N is even and $B_{\frac{N}{2}} = 0$.
- the *mid-tread quantizer:* N is odd and $B_{\frac{N-1}{2}} = -\frac{\Delta}{2}$. For this quantizer, $Q(x) = 0$ when $x \in [-\frac{\Delta}{2}, \frac{\Delta}{2}]$

The reason for the names "mid-rise" and "mid-tread" has to do with the geometrical picture of the functions $Q()$. For the mid-rise quantizer, the origin is in the middle of a rising edge. For the mid-tread quantizer, the origin is in the middle of a flat region (tread, i.e. where you step). [Note that both types of functions have odd symmetry about the origin, that is, $Q(-x) = -Q(x)$.]

For the midrise quantizer, the values of $Q(x)$ are $\ldots, -\frac{5}{2}\Delta, -\frac{3}{2}\Delta, -\frac{1}{2}\Delta, \frac{1}{2}\Delta, \frac{3}{2}\Delta, \frac{5}{2}\Delta, \ldots$, and the values of B_i are multiples of Δ. For the midtread quantizer, the values of $Q(x)$ are multiples of Δ, and the values of the B_i are $\ldots, -\frac{5}{2}\Delta, -\frac{3}{2}\Delta, -\frac{1}{2}\Delta, \frac{1}{2}\Delta, \frac{3}{2}\Delta, \frac{5}{2}\Delta, \ldots$.





Distorsion vs. Rate (extra material – not on final exam)

A more general setting is to allow ourselves to choose boundaries B_i and reconstruction levels y_i without requiring ourselves to have uniform differences Δ between the boundaries, where $y_i = Q(x)$, when $B_i < x \leq B_{i+1}$. This is the setting of *non-uniform* quantization. Let's suppose that we have N quantization levels as before and that $B_i < y_i < B_{i+1}$ for all i. Now we ask how to choose these B_i and y_i values.

Suppose we choose the B_i. Then we would like to choose the y_i such that we minimize the *expected quantization error*, also known as the *distorsion*, namely:

$$\mathcal{D} = \sum_{i=0}^{N-1} \int_{B_i}^{B_{i+1}} (y_i - x)^2 p(x) dx.$$

We don't necessarily need to use the squared error, but this is common. We could also pose this the opposite way, namely choose the set of y_i values, and then try to solve for the B_i values that minimize this above mean squared error. Or we could use some iterative scheme where we first choose the y_i (say uniformly spaced) and then solve the B_i; then for these B_i fixed, we could solve for the y_i that minimizes the error; etc. Regardless of how we do this, the problem we are solving is to minimize the error or distorsion.

Let's generalize the scheme in another way, by allowing ourselves to use a variable length code to encode the quantization intervals, instead of a fixed length code (which is what we have been implicitly assuming up to now – namely, we have N intervals and the encoder uses $\log N$ bits to encode the interval i).

If we use λ_i bits for interval i, where $i \in 1, \ldots, N$, then average number of bits used is the *rate*

$$\overline{\lambda} = \sum_{i=1}^{N} \int_{B_i}^{B_{i+1}} \lambda_i \; p(x) dx.$$

We would like to choose boundaries B_i and codeword lengths λ_i that minimize this rate.

Notice that we can make the rate arbitrarily small by having one interval cover almost all possible values of X and using just a one bit codeword for this case (say codeword 0), and the remaining $N - 1$ intervals cover nearly none of the values of X and using two or more bits for these values of X (say codeword 1*). However, although such a scheme would give a low *rate*, it would give a high *distorsion.*

Similarly, though this is not as obvious, reducing the distorsion does not necessarily allow us to choose a code with the lowest rate (since the former reduces a mean squared error and is independent of codeword length) whereas the latter depends fundamentally on how the codes are chosen.