Coding with distortion

We have a signal $x_n,\ n = 1 \ldots N$ to code. The alphabet is a subset of the real numbers $\mathcal{A} \subseteq \mathbb{R}$. The alphabet can be continuous.

If we don't have the demand that the decoded signal should be exactly the same as the original signal we can get a lower data rate than if we have lossless coding. Typically the signal is described using a smaller alphabet than the original signal uses (quantization).

In the case where the original alphabet is continuous, in general an infinite number of bits is required to describe the signal losslessly.

The more bits that are used, the closer to the original signal the decoded signal $\hat{x}_n$ will be.

Distortion measure

We need a measure of how much error we have in the decoded signal, the so called *distortion*.

The most common measure is a quadratic error measure, combined with averaging over the whole sequence

$$D = \frac{1}{N} \sum_{n=1}^{N} (x_n - \hat{x}_n)^2$$

This is the *mean square error* of the decoded sequence.

Distortion measure, cont.

Often we want to consider the distortion (or noise power) relative to the signal power, the so called *signal to noise ratio* (SNR)

$$\sigma_x^2 = \frac{1}{N} \sum_{n=1}^{N} x_n^2$$

$$\text{SNR} = \frac{\sigma_x^2}{D}$$

SNR is usually expressed in dB

$$\text{SNR} = 10 \cdot \log_{10} \frac{\sigma_x^2}{D}$$

PSNR

When coding still images and video we usually use the *peak-to-peak signal to noise ratio* (PSNR)

$$\text{PSNR} = 10 \cdot \log_{10} \frac{x_{pp}^2}{D}$$

where x_{pp} is the difference between the maximum and minum values of the signal.

For example, if the data to be coded is a grayscale image quantized to 8 bits, the signal can assume values between 0 and 255. The PSNR is then

$$\text{PSNR} = 10 \cdot \log_{10} \frac{255^2}{D}$$

Random signal models

A signal can be modelled as an amplitude continuous stationary random process X_n, with distribution function $F_X(x)$ and density function $f_X(x)$.

$$F_X(x) = Pr(X \leq x)$$

$$f_X(x) = \frac{d}{dx} F_X(x)$$

$$f_X(x) \geq 0 \ , \quad \forall x$$

$$\int_{-\infty}^{\infty} f_X(x) dx = 1$$

$$Pr(a \leq X \leq b) = F_X(b) - F_X(a) = \int_a^b f_X(x) dx$$

Random signal models, cont.

Mean value

$$m_X = E\{X_n\} = \int_{-\infty}^{\infty} x \cdot f_X(x) dx$$

Quadratic mean value

$$E\{X_n^2\} = \int_{-\infty}^{\infty} x^2 \cdot f_X(x) dx$$

Variance

$$\sigma_X^2 = E\{(X_n - m_x)^2\} = E\{X_n^2\} - m_x^2$$

In most of our cases we will use signal models with mean value 0. In those cases the variance is equal to the quadratic mean value.

The variance (or rather the quadratic mean value) is a measure of the signal power.

Common distributions

Uniform distribution

$$f_X(x) = \begin{cases} \frac{1}{b-a} & a \leq x \leq b \\ 0 & \text{otherwise} \end{cases}$$

Mean value $m = \frac{a+b}{2}$, variance $\sigma^2 = \frac{(b-a)^2}{12}$

Gaussian distribution (normal distribution)

$$f_X(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-m)^2}{2\sigma^2}}$$

Laplace distribution

$$f_X(x) = \frac{1}{\sqrt{2}\sigma} e^{-\frac{\sqrt{2}|x-m|}{\sigma}}$$

Random signal models, cont.

The dependence of the signal value in two times instances n och m is given by the twodimensional density function $f_{X_n X_m}(x_n, x_m)$.

If we can write this as a product $f_X(x_n) \cdot f_X(x_m)$ we say that the signal in the two time instances are *independent*.

A signal where all time instances are independent of each other is a *memoryless* signal or a *white* signal.

In most cases we will describe the dependence using the *correlation* $E\{X_n \cdot X_m\}$.

If $E\{X_n \cdot X_m\} = E\{X_n\} \cdot E\{X_m\}$ we say that the signal in the two time instances are *uncorrelated*. Independent signals are uncorrelated, but the reverse is not necessarily true.

Memory sources

Markov source of order k

$$f(x_n|x_{n-1}x_{n-2}\ldots) = f(x_n|x_{n-1}\ldots x_{n-k})$$

Linear models, ϵ_n white (memoryless) noise.
AR(N)

$$x_n = \sum_{i=1}^{N} a_i \cdot x_{n-i} + \epsilon_n$$

MA(M)

$$x_n = \sum_{j=1}^{M} b_i \cdot \epsilon_{n-j} + \epsilon_n$$

ARMA(N, M)

$$x_n = \sum_{i=1}^{N} a_i \cdot x_{n-i} + \sum_{j=1}^{M} b_i \cdot \epsilon_{n-j} + \epsilon_n$$

Random signal models, cont.

The correlation properties of the signal is usually expressed using the *auto correlation function*, which for a stationary process is given by

$$R_{XX}(k) = E\{X_n X_{n+k}\}$$

The auto correlation function is symmetric: $R_{XX}(-k) = R_{XX}(k)$.

We also have: $|R_{XX}(k)| \leq R_{XX}(0) = E\{X_n^2\}$.

For a memoryless (white) process we have

$$R_{XX}(k) = \sigma_X^2 \cdot \delta(k) = \begin{cases} \sigma_X^2 & k = 0 \\ 0 & \text{otherwise} \end{cases}$$

For an AR(1) process we have

$$R_{XX}(k) = a^{|k|} \cdot \sigma_X^2 \qquad (|a| < 1)$$

Multidimensional signals

The auto correlation function can of course also be defined for multidimensional signals. For instance, for a twodimensional stationary random process $X_{i,j}$ the auto correlation function is given by

$$R_{XX}(k,l) = E\{X_{i,j}X_{i+k,j+l}\}$$

The auto correlation function is symmetric: $R_{XX}(-k,-l) = R_{XX}(k,l)$

We also have: $|R_{XX}(k,l)| \leq R_{XX}(0,0) = E\{X_{i,j}^2\}$

Random signal models, cont.

For a random signal X_n that is coded and then decoded to $\hat{X}_n$, the distortion is given by

$$D = E\{(X - \hat{X})^2\} = \int_{-\infty}^{\infty} (x - \hat{x})^2 f_X(x) dx$$

The signal power is (given mean zero)

$$E\{X^2\} = \sigma_X^2 - (E\{X\})^2 = \sigma_X^2$$

and SNR as before

$$\text{SNR} = 10 \cdot \log_{10} \frac{\sigma_X^2}{D}$$

Theoretical limit

The rate-distortion function $R(D)$ for a source gives the theoretically lowest rate R we can use to code the source, on the condition that the maximum allowed distortion is D. Compare to the entropy limit for source coding.

Example: White gaussian process with variance σ^2

$$R(D) = \begin{cases} \frac{1}{2} \log \frac{\sigma^2}{D} & 0 < D \leq \sigma^2 \\ 0 & \text{otherwise} \end{cases}$$

Ie, if we allow a distortion that is larger than the variance of the process, we don't need to transmit any bits at all. The decoder can just set the decoded signal equal to the mean value at each time instance, which will give a distortion equal to the variance.
We can also se that $R \to \infty$ when $D \to 0$

Gaussian source with memory

For gaussian sources with memory, the rate-distortion function can be calculated from the power spectral density.

$$\Phi(\theta) = \mathcal{F}\{R_{XX}(k)\} = \sum_{k=-\infty}^{\infty} R_{XX}(k) \cdot e^{-j2\pi\theta k}$$

The rate-distortion function is given by

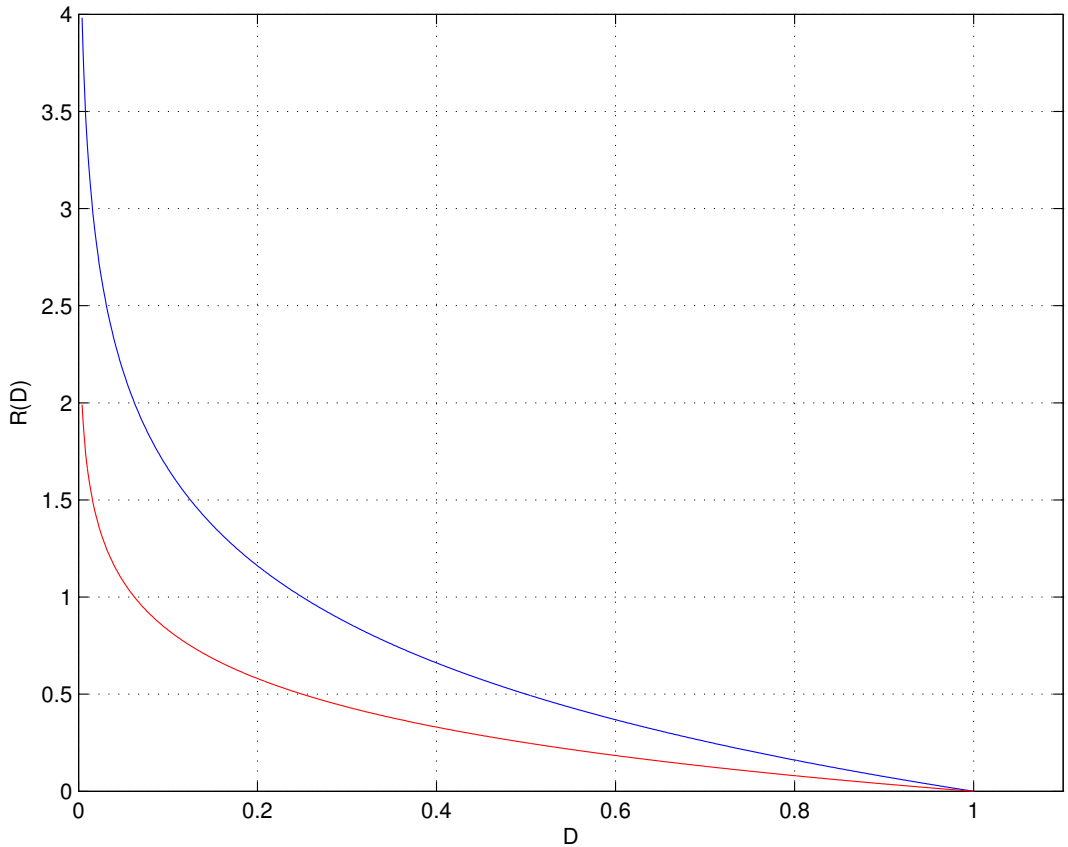
$$R(D) = \int_{-1/2}^{1/2} \max\{\frac{1}{2} \log \frac{\Phi(\theta)}{\lambda}, 0\} \; d\theta$$

where

$$D = \int_{-1/2}^{1/2} \min\{\lambda, \Phi(\theta)\} \; d\theta$$

The integration can of course be done over any interval of size 1, since the power spectral density is a periodic function.

Gaussian sources



$R(D)$ for an ideally bandlimited gaussian source (red), compared to the $R(D)$ for a memoryless/white gaussian source (blue). Both sources have variance 1. As D tends towards 0, $R(D)$ tends towards infinity for both sources.

Quantization

Mapping from a continuous alphabet to a discrete alphabet (or mapping from a large discrete alphabet to a smaller one). After quantization we have a discrete signal, on which we can use our source coding methods (Huffman, arithmetic coding, et c.)

A general M level quantizer is specified by $M+1$ *decision borders* $b_i \; ; \; i = 0 \ldots M$ and M *reconstruction levels (or reconstruction points)* $y_i \; ; \; i = 1 \ldots M$.

The quantization operator $Q(x)$ is given by

$$Q(x) = y_i \quad \text{if} \quad b_{i-1} < x \leq b_i$$

And the reconstructed signal is thus

$$\hat{x}_n = Q(x_n)$$

Quantization

Sometimes it can be useful to see quantization and reconstruction as two separate operations instead of just one operation.

Quantization: $x \to j$ such that $b_{j-1} < x \leq b_j$

Reconstruction: $\hat{x} = y_j$

A sequence of x thus gives a sequence of indices j that can then be coded by a source coder

The receiver decodes the index sequence and the maps the indices to the corresponding reconstruction points.

Quantization, cont.

Given a random signal model the distortion is

$$\begin{aligned} D &= E\{(X-\hat{X})^2\} = \\ &= \int_{-\infty}^{\infty} (x-Q(x))^2 f_X(x)dx = \\ &= \sum_{i=1}^{M} \int_{b_{i-1}}^{b_i} (x-y_i)^2 f_X(x)dx \end{aligned}$$

If no special source coding is used, ie if we just code the quantized signal using a fixed length code, the rate is

$$R = \lceil \log_2 M \rceil$$

Uniform quantization

The distance between two reconstruction points is constant

$$y_j - y_{j-1} = \Delta$$

Δ is the *stepsize* of the quantizer.

The reconstruction points are in the middle of their intervals, which means that all decision regions (apart from the end intervals in some cases) also are of the same size

$$b_i - b_{i-1} = \Delta$$

Uniform quantization, cont.

To simplify the calculations we can assume that the number of reconstruction points is given by $M = 2^R$ and that the quantizer is symmetric around the origin. The following results can easily be generalized to arbitrary M.

The reconstruction point belonging to the interval $[(j-1)\Delta, j\Delta]$ is

$$y_j = \frac{2j-1}{2}\Delta$$

The simplest case is when the input distribution is uniform on the interval $[-A, A]$:

$$\Delta = \frac{2A}{M}$$

Uniform quantization, cont.

The distortion for uniform quantization of a uniform distribution:

$$D = \sum_{i=-M/2+1}^{M/2} \int_{(i-1)\Delta}^{i\Delta} (x - \frac{2i-1}{2}\Delta)^2 \frac{1}{2A} dx = M \cdot \frac{1}{2A} \cdot \frac{\Delta^3}{12} = \frac{\Delta^2}{12}$$

$$\sigma_X^2 = \frac{(2A)^2}{12} = \frac{\Delta^2 M^2}{12}$$

$$\begin{aligned} \text{SNR} &= 10 \cdot \log_{10} \frac{\sigma_X^2}{D} = 10 \cdot \log_{10} M^2 = \\ &= 10 \cdot \log_{10} 2^{2R} = 20 \cdot R \cdot \log_{10} 2 \approx 6.02 \cdot R \end{aligned}$$

For every bit added to the quantizer (ie for every doubling of the number of reconstruction points) we will get approximately 6 dB higher SNR.

Uniform quantization, cont.

For unlimited distributions (eg a gaussian distribution) the two end intervals will be infinitely large (in the calculations below we assume that that M is even and that the quantizer is symmetric around the origin).

$$
\begin{aligned}
D &= \sum_{i=-M/2+1}^{M/2} \int_{(i-1)\Delta}^{i\Delta} (x - \frac{2i-1}{2}\Delta)^2 f_X(x)dx + \\
&+ \int_{(M/2)\Delta}^{\infty} (x - \frac{M-1}{2}\Delta)^2 f_X(x)dx + \\
&+ \int_{-\infty}^{-(M/2)\Delta} (x - \frac{-M+1}{2}\Delta)^2 f_X(x)dx
\end{aligned}
$$

The last two terms are called the *overload distortion* of the quantizer.

Uniform quantization, cont.

To find the best choice of Δ (the one that minimizes the distortion) we have to solve

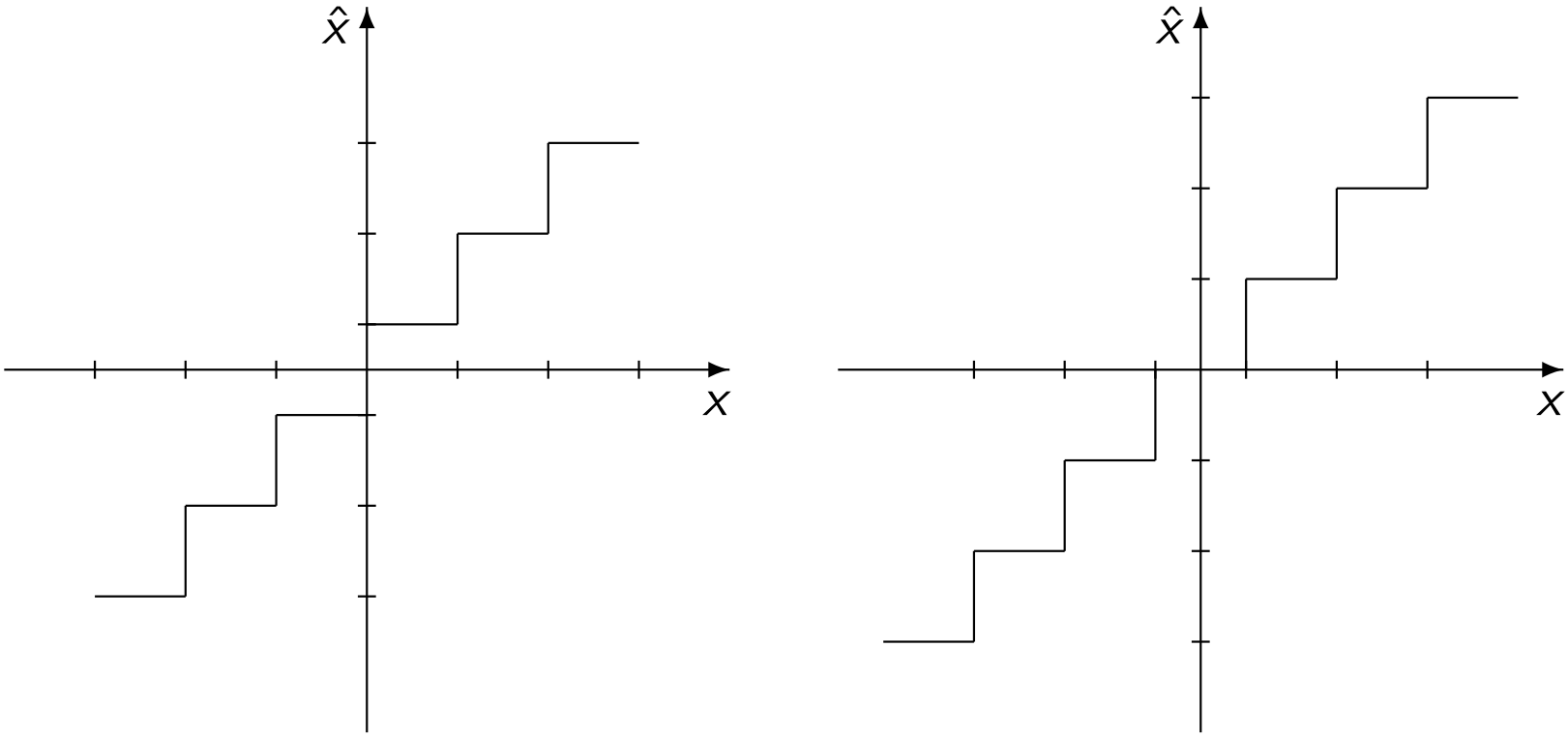
$$\frac{\partial}{\partial \Delta} D = 0$$

which in the general case is a hard problem. Normally we will have to find a numeric solution.

If the number of quantization levels M is large and Δ is chosen so that the overload distortion is small compared to the total distortion, the distortion is approximately

$$D \approx \frac{\Delta^2}{12}$$

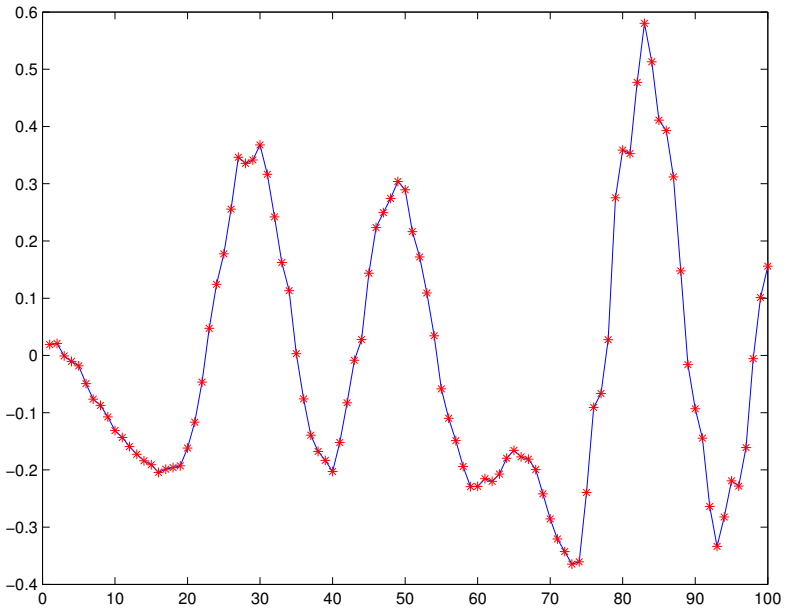
Midrise vs midtread



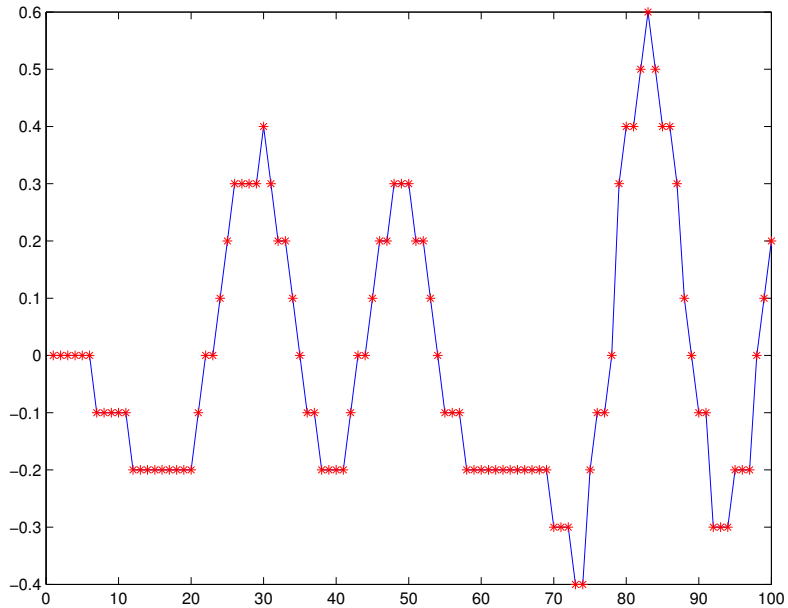
When coding audio and image data we usually want to use a quantizer that has a reconstruction level in 0, ie a midtread quantizer.

Uniform quantization, example

Uniform quantization (midtread) of a speech signal



Original signal



Quantized using $\Delta = 0.1$

Measured distortion: $D \approx 0.0008517$
(Compare to $\Delta^2/12 \approx 0.0008333$)

Lloyd-Max quantization

How should we chose decision borders and reconstruction points to minimize the distortion? The answer will of course depend on the distribution of the signal.

For a general quantizer we have

$$D = \sum_{i=1}^{M} \int_{b_{i-1}}^{b_i} (x - y_i)^2 f_X(x) dx$$

We want to find the the quantizer that minimizes the distortion D.

$$\frac{\partial}{\partial y_j} D = 0 \;\Rightarrow\; y_j = \frac{\int_{b_{j-1}}^{b_j} x \cdot f_X(x) dx}{\int_{b_{j-1}}^{b_j} f_X(x) dx}$$

The optimal placement of the reconstruction points is thus in the centroid of the probability mass in each interval.

Lloyd-Max quantization, cont.

$$\frac{\partial}{\partial b_j} D = 0 \;\Rightarrow\; b_j = \frac{y_{j+1} + y_j}{2}$$

The optimal placement of the decision borders is thus at the midpoints between the reconstruction points, ie we should always quantize to the closest reconstruction point.

Note that these demands are necessary but not sufficient.

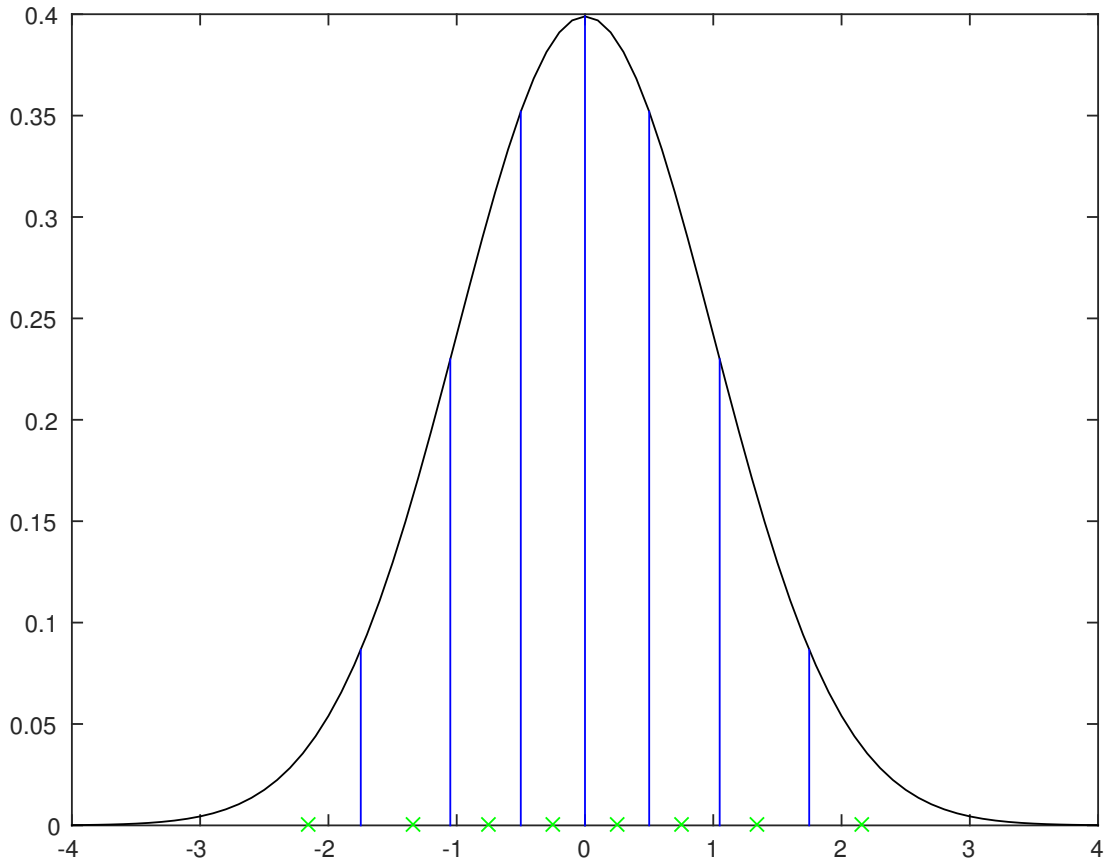
Also note that y_j depends on b_{j-1} and b_j and that b_j depends on y_{j+1} and y_j. Usually we can only find closed solutions for simple distributions and for a small number of reconstruction points.

If we can't find a closed solution, we have to find the solution numerically. One way of doing this is using Lloyd's algorithm.

Lloyd's algorithm

1. Start with a set of reconstruction points $y_i^{(0)},\ i = 1 \dots M$. Set $k = 0$, $D^{(-1)} = \infty$ and choose a threshold ϵ.
2. Calculate optimal decision borders $b_j^{(k)} = \dfrac{y_{j+1}^{(k)} + y_j^{(k)}}{2}$
3. Calculate the distortion $D^{(k)} = \sum_{i=1}^{M} \int_{b_{i-1}^{(k)}}^{b_i^{(k)}} (x - y_i^{(k)})^2 f(x) dx$
4. If $D^{(k-1)} - D^{(k)} < \epsilon$ stop, otherwise continue
5. $k = k + 1$. Calculate new optimal reconstruction points
$$y_j^{(k)} = \frac{\int_{b_{j-1}^{(k-1)}}^{b_j^{(k-1)}} x \cdot f(x) dx}{\int_{b_{j-1}^{(k-1)}}^{b_j^{(k-1)}} f(x) dx}$$
6. Repeat from 2

Example



Lloyd-Max quantizer for a Gaussian distribution. $R = 3$, ie $M = 2^3 = 8$.

Compander quantization

Compressor function: $c(x)$
Expander function: $c^{-1}(x)$

Quantize $c(x)$ instead of x. The quantization is uniform. The receiver applies the expander function on received data.
$c(x)$ is usually chosen such that the quantization is finer for small values of x.

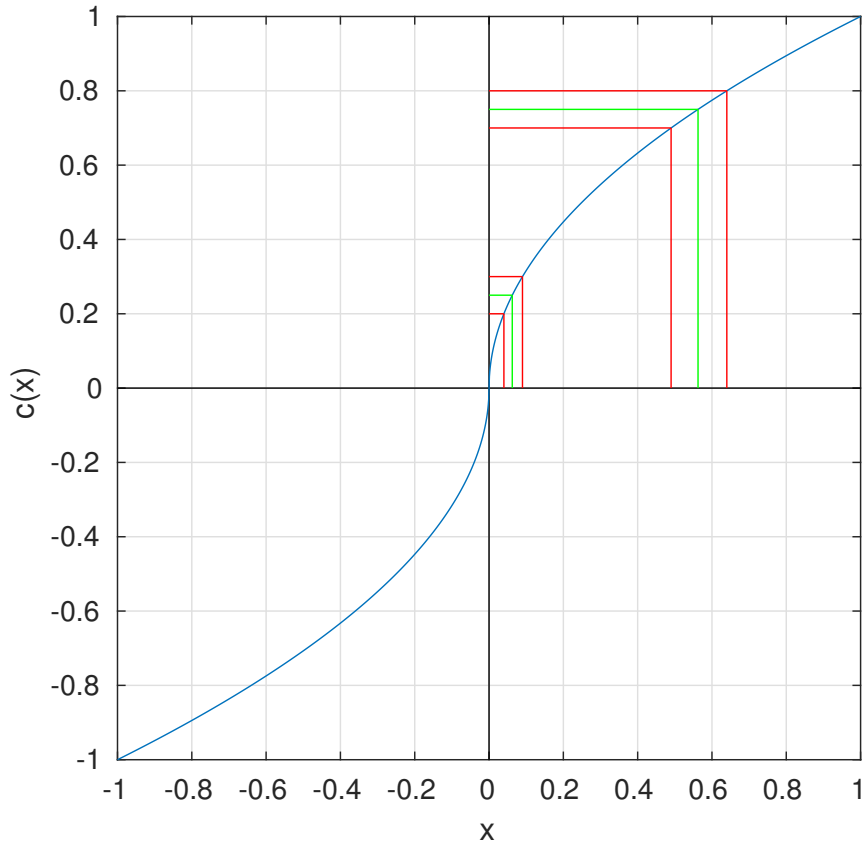
Example of compander: μ-law

$$c(x) = x_{\max} \frac{\ln(1 + \mu \frac{|x|}{x_{\max}})}{\ln(1 + \mu)} \cdot \operatorname{sgn}(x)$$

$$c^{-1}(x) = \frac{x_{\max}}{\mu}((1 + \mu)^{\frac{|x|}{x_{\max}}} - 1) \cdot \operatorname{sgn}(x)$$

used in the american wired telephone network for coding of speech signals using $\mu = 255$. For the european telephone network there is a similar standard called *A-law*.

Compander example



Compressor function: $c(x) = \text{sgn}(x) \cdot |x|^{0.5}$
Uniform quantization (in the example $\Delta = 0.1$) of $c(x)$ corresponds to a non-uniform quantization of x, with smaller quantization intervals closer to the origin.

Quantization using source coding

The probability $P(j)$ of being in interval j is

$$P(j) = \int_{b_{j-1}}^{b_j} f_X(x)dx$$

In the general case these probabilities are different for different intervals. We could thus get a lower rate than $\log M$ by using some form of source coding.

Finding the optimal quantizer given an allowed rate R after source coding is a hard problem. However, it can be shown that for sufficiently large R (fine quantization) the optimal quantizer is a uniform quantization. Thus, if we are using some form of source coding, it is enough to use the simplest form of quantization.

Fine quantization

When we have *fine quantization*, ie when the number of quantization levels is large, the distortion is approximatively given by

$$D \approx c \cdot \sigma_X^2 \cdot 2^{-2R}$$

where σ_X^2 is the signal variance, R is the rate and c is a constant depending on the type of quantization and the distribution of the signal.

Gaussian distribution, Lloyd-Max quantization:

$$c = \frac{\pi\sqrt{3}}{2}$$

Gaussian distribution, uniform quantization, perfect source coding ($R = H(\hat{X})$):

$$c = \frac{\pi e}{6}$$

Fine quantization

For fine quantization we have the approximations

$$\begin{cases} D \approx \frac{1}{12} \int_{-\infty}^{\infty} \Delta^2(x) f(x) dx \\ M \approx \int_{-\infty}^{\infty} \frac{1}{\Delta(x)} dx \end{cases}$$

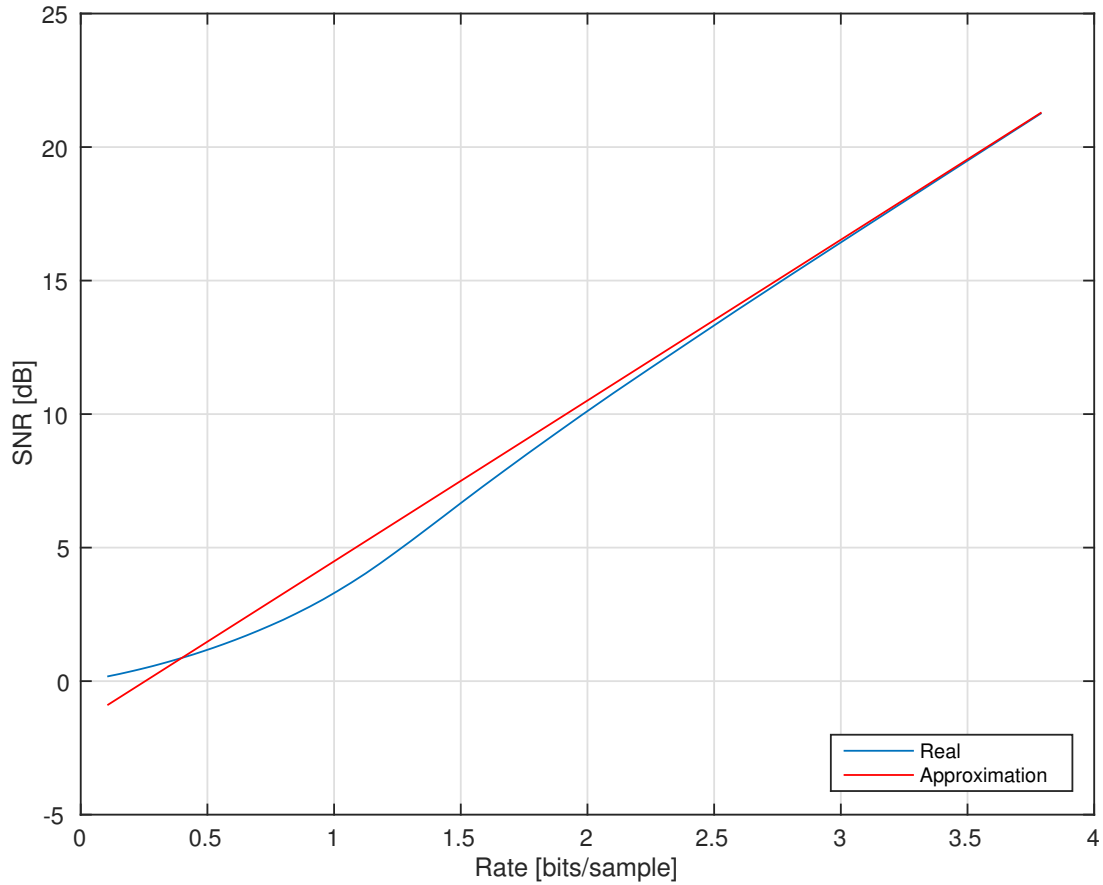
where $\Delta(x)$ is a function describing the size of the quantization interval at x and M is the resulting number of reconstruction points.

For fine Lloyd-Max quantization, we should choose

$$\Delta(x) = k \cdot (f(x))^{-\frac{1}{3}}$$

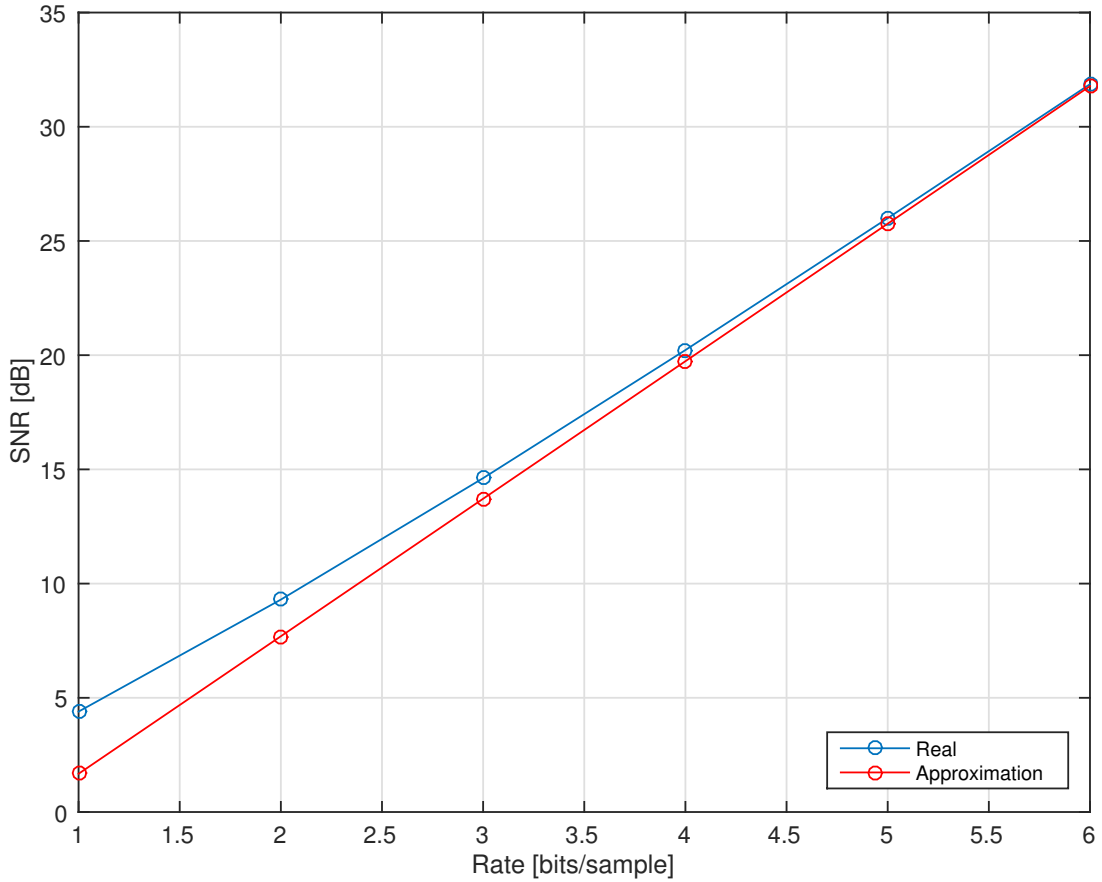
and the resulting rate will be $R = \log_2 M$.

Approximation vs real values



Uniform quantization of Gaussian signal, followed by perfect source coding ($R = H(\hat{X})$). Real values compared to the approximation $D \approx \frac{\pi e}{6} \cdot \sigma_X^2 \cdot 2^{-2R}$.

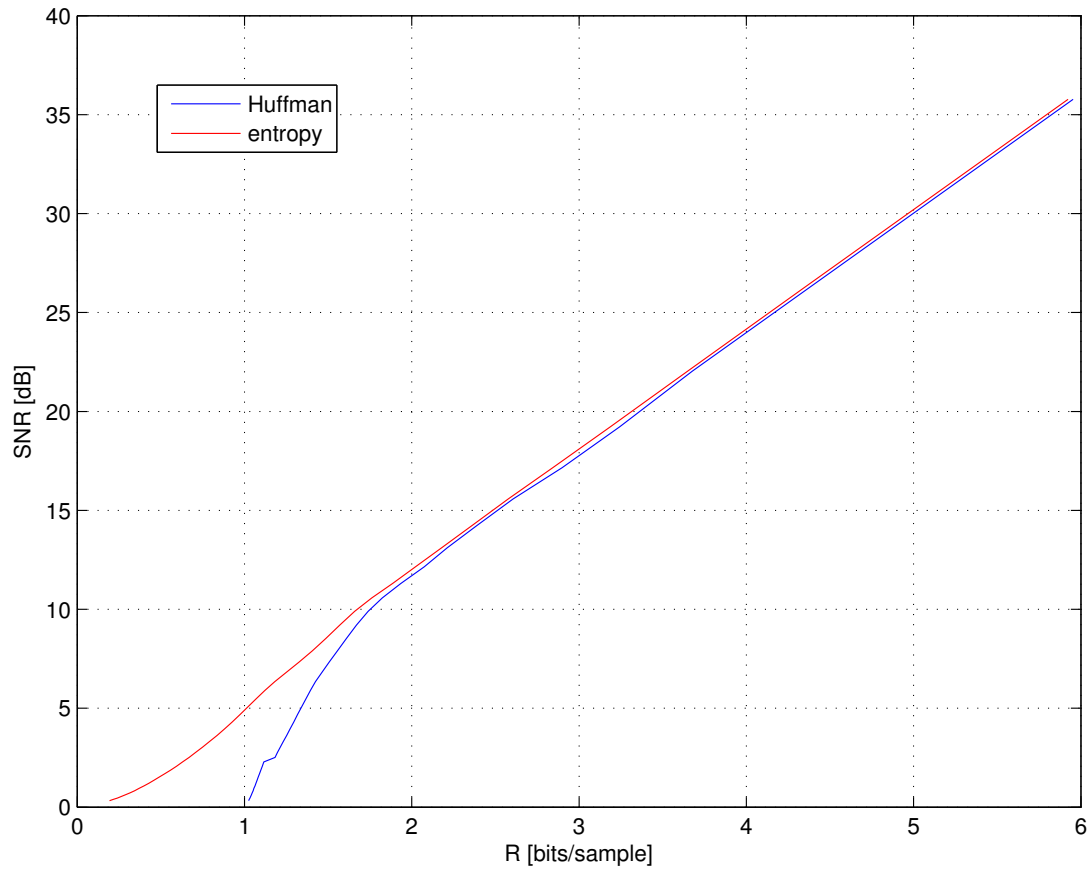
Approximation vs real values



Lloyd-Max quantization of Gaussian signal. Real values compared to the approximation $D \approx \frac{\pi\sqrt{3}}{2} \cdot \sigma_X^2 \cdot 2^{-2R}$.

Real world signal

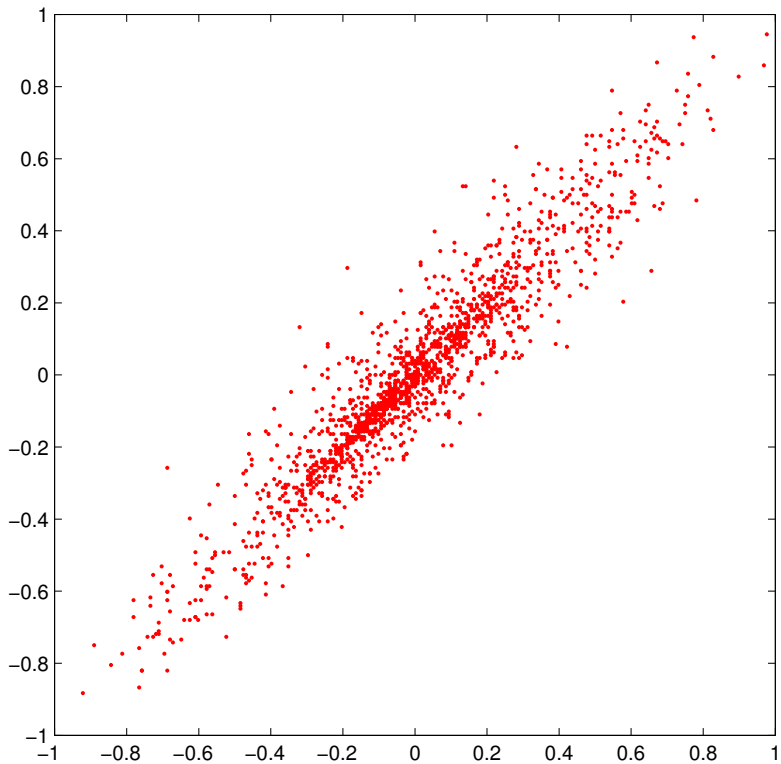
A mono version of the music file `heyhey.wav` is coded using a uniform quantizer (midtread), followed by Huffman coding. The rate is varied by varying the quantizer stepsize. No limitation of the number of levels is done. For comparison, we have also estimated the entropy of the quantized signal.



Vector quantization

Consecutive samples in a signal are often strongly correlated.

Example: 4000 samples from a speech signal, plot $[x_i, x_{i+1}]$



Vector quantization, cont.

In order to utilize the dependence between samples we can use some form of source coding that uses the memory of the signal, eg extended Huffman coding or arithmetic coding where we let the interval division depend on previous symbols.

We can also use the correlation between samples directly in the quantizer, by quantizing several samples at once, *vector quantization*.

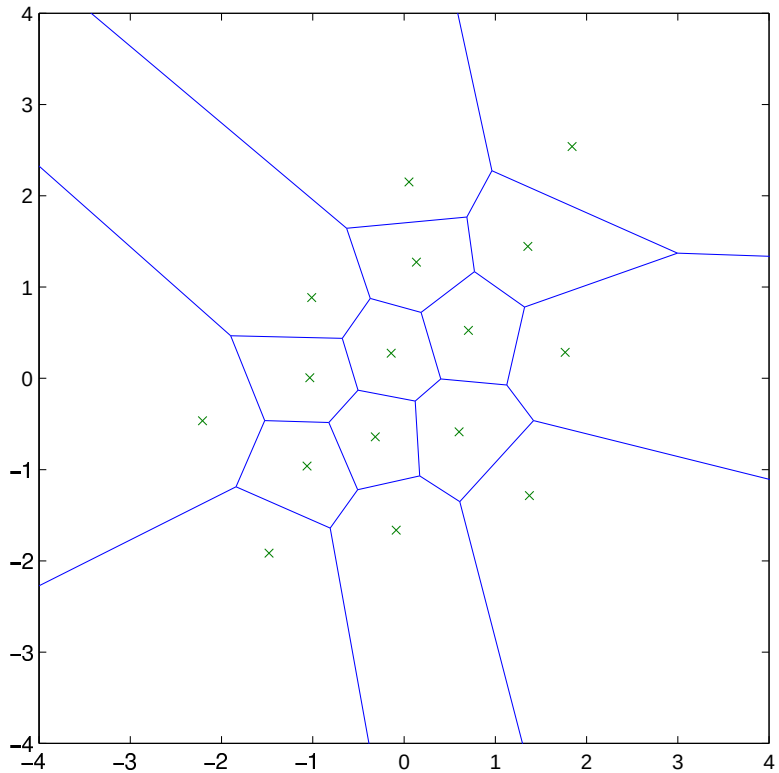
View L samples from the source as a L-dimensional vector.

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_L \end{pmatrix}$$

Vector quantization, cont.

The quantizer is defined by its reconstruction vectors $\mathbf{y}_i$ and decision regions V_i.

Example: In two dimensions it can look like



Vector quantization, cont.

The set of reconstruction points $\{\mathbf{y}_i\}, i = 1 \ldots M$ is usually called the *codebook*.

Distortion

$$D = \frac{1}{L} \sum_{i=1}^{M} \int_{V_i} |\mathbf{x} - \mathbf{y}_i|^2 \cdot f(\mathbf{x}) \; d\mathbf{x}$$

Compare this to the distortion when doing scalar quantization

$$D = \sum_{i=1}^{M} \int_{b_{i-1}}^{b_i} (x - y_i)^2 f(x) dx$$

A scalar quantizer is a onedimensional vector quantizer.

Data rate

To code the M reconstruction points using a fixed length code we need $\lceil \log M \rceil$ bits. Since we're coding L samples at a time, the resulting rate is

$$R = \frac{\lceil \log M \rceil}{L} \quad \text{[bits/sample]}$$

Alternatively, given dimension L and rate R

$$M = 2^{RL}$$

It is of course possible to use some kind of source coding method when doing vector quantization, but that's usually not done.

Storage and time

We need to store the codebook both on the coder and the decoder side. It might also have to be transmitted as side information. If we are using L dimensions, have a rate of R bits per sample and each element of the vectors is stored using b bits, we need

$$2^{RL} \cdot L \cdot b$$

bits to store the whole codebook. The required storage space thus grows very quickly when we increase the dimension.

If there is no structure in the codebook we have to compare each vector that we quantize with *all* the vectors in the codebook in order to find the closest one. When the dimension L is large this will take a lot of time.

Lloyd-Max in multiple dimensions

We want to find a quantizer that minimizes the distortion.
Necessary conditions for minimal distortion:

- The borders between decision regions should be in the middle between the reconstruction points, ie we should quantize to the closest reconstruction point (ie the decision regions are the Voronoi regions of the reconstruction points).
- The reconstruction points should be located in the *centroids* of the decision regions

$$\mathbf{y}_i = \frac{\int_{V_i} \mathbf{x} \cdot f(\mathbf{x}) d\mathbf{x}}{\int_{V_i} f(\mathbf{x}) d\mathbf{x}}$$

Compare this to Lloyd-Max quantization in one dimension.

Lloyd's generalized algorithm

1. Begin with a starting codebook $\mathbf{y}_i^{(0)}$, $i = 1 \ldots M$. Let $k = 0$, $D^{(-1)} = \infty$ and choose a threshold ϵ.
2. Determine optimal decision regions

$$V_i^{(k)} = \{\mathbf{x} : |\mathbf{x} - \mathbf{y}_i|^2 < |\mathbf{x} - \mathbf{y}_j|^2 \ \forall j \neq i\}$$

3. Calculate the distortion

$$D^{(k)} = \sum_{i=1}^{M} \int_{V_i^{(k)}} |\mathbf{x} - \mathbf{y}_i^{(k)}|^2 f(\mathbf{x}) d\mathbf{x}$$

4. If $D^{(k-1)} - D^{(k)} < \epsilon$ stop, otherwise continue.
5. $k = k + 1$. Determine new optimal reconstruction points as the centroids of each $V_i^{(k-1)}$.
6. Repeat from 2.

Training data

Usually we don't know the probability density function $f(\mathbf{x})$ of the source. One way would be to estimate the density function from a long sequence from the source (training data).

Instead of doing this, Lloyd's algorithm can be modified to use the training data directly.

This variant of the algorithm is usually called the *LBG algorithm* or *K-means*.

The LBG algorithm

1. Begin with a starting codebook $\mathbf{y}_i^{(0)},\ i = 1 \ldots M$ and a set of training vectors $\mathbf{x}_n,\ n = 1 \ldots N$. Let $k = 0$, $D^{(-1)} = \infty$ and choose a threshold ϵ.
2. Determine optimal decision regions

$$V_i^{(k)} = \{\mathbf{x}_n : |\mathbf{x_n} - \mathbf{y}_i|^2 < |\mathbf{x_n} - \mathbf{y}_j|^2 \ \forall j \neq i\}$$

3. Calculate the distortion $D^{(k)} = \sum_{i=1}^{M} \sum_{\mathbf{x}_n \in V_i^{(k)}} |\mathbf{x_n} - \mathbf{y}_i^{(k)}|^2$
4. If $D^{(k-1)} - D^{(k)} < \epsilon$ stop, otherwise continue.
5. $k = k + 1$. Determine new optimal reconstruction points as the average of all vectors in each $V_i^{(k-1)}$.
6. Repeat from 2.

How to choose the starting codebook

Depending on how we choose the starting codebook we can get different resulting codebooks. A few variants:

- Choose M arbitrary vectors.
- Choose M vectors from the training data.
- Generate several random starting codebooks and choose the one that gives the lowest distortion.
- PNN (Pairwise Nearest Neighbour).
  Start with each training vector as a reconstruction point. In each step remove the two vectors that are closest to each other and replace them with the average of the two vectors. Repeat until we have M vectors.

Variant of LBG: Splitting

1. Start with a single reconstruction point given by the average of all training vectors.
2. Double the size of the codebook by adding a small perturbation vector to each reconstruction vector.
3. Optimize the codebook using the LBG algorithm.
4. If we have M vectors in the codebook we are finished, otherwise move to step 2.

Empty regions

What do we do if a region becomes empty during a step in the LBG algorithm?

Replace the reconstruction vector that has an empty region with a new vector. A few variants:

1. Choose the new reconstruction point randomly from the region that has the highest number of training data.
2. Choose the new reconstruction point randomly from the region that has the largest distortion.
3. Optimize a two level quantizer in the region that has the largest distortion.

Method 3 is more computationally intensive, but it doesn't give any benefits compared to the other methods.

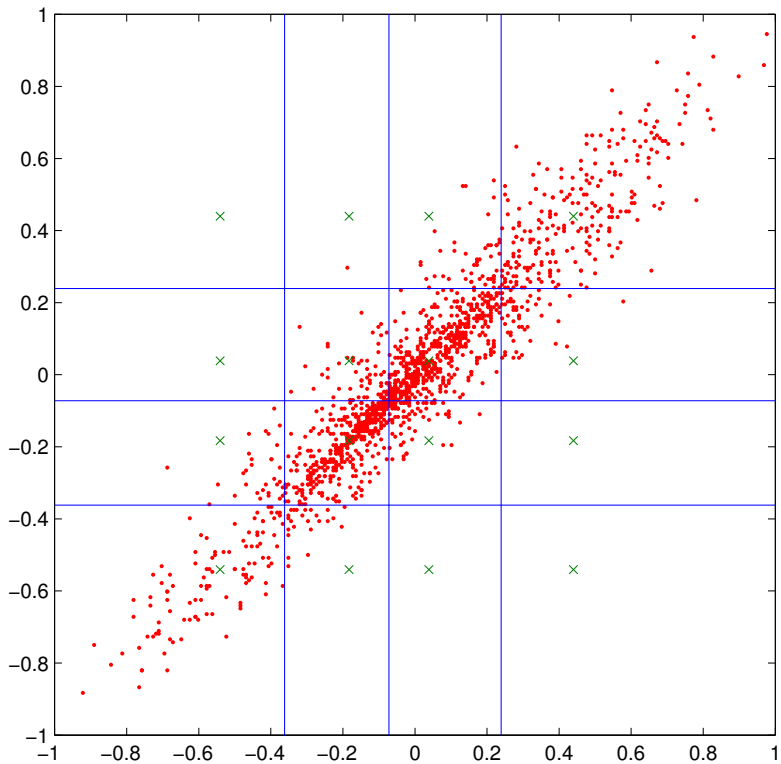
Advantages and disadvantages

Advantages and disadvantages with vector quantization compared to scalar quantization.

+ Can utilize the memory of the source.
+ The distortion at a given rate will always be lower when increasing the number of dimensions, even for a memoryless source.
– Both the storage space and the time needed to perform the quantization grows faster than exponentially with the number of dimensions. Since there is no structure to the codebook (in the general case) we will have to compare each signal vector with every reconstruction vector in the codebook to find the closest one.

LBG, example

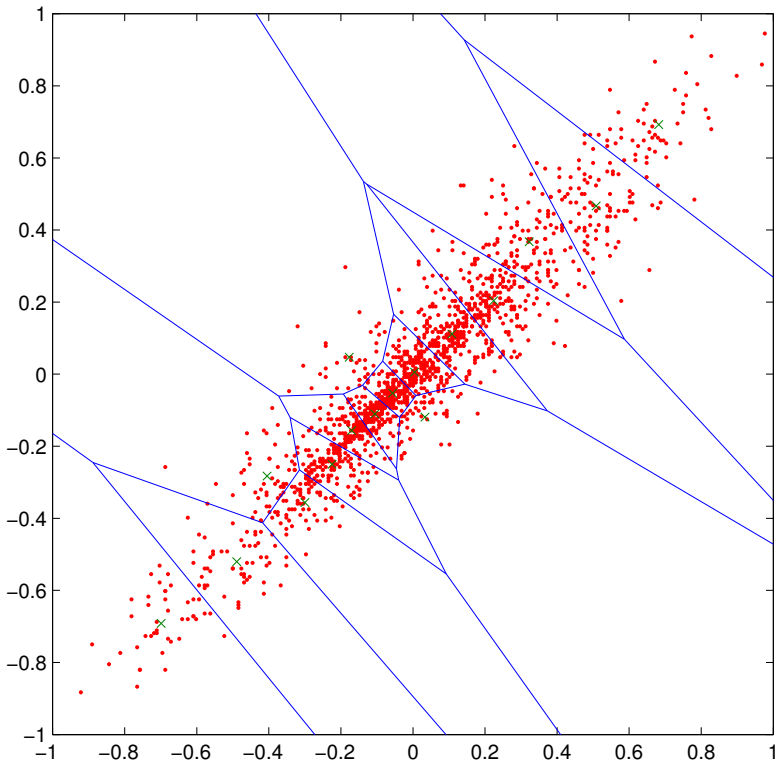
We optimize a scalar quantizer with rate 2 bits/sample (ie $2^2 = 4$ levels) for our speech signal. If we view this quantizer in two dimensions, where we quantize each dimension separately, the decision regions and reconstruction points look like



SNR is approximately 8.6 dB.

LBG, example

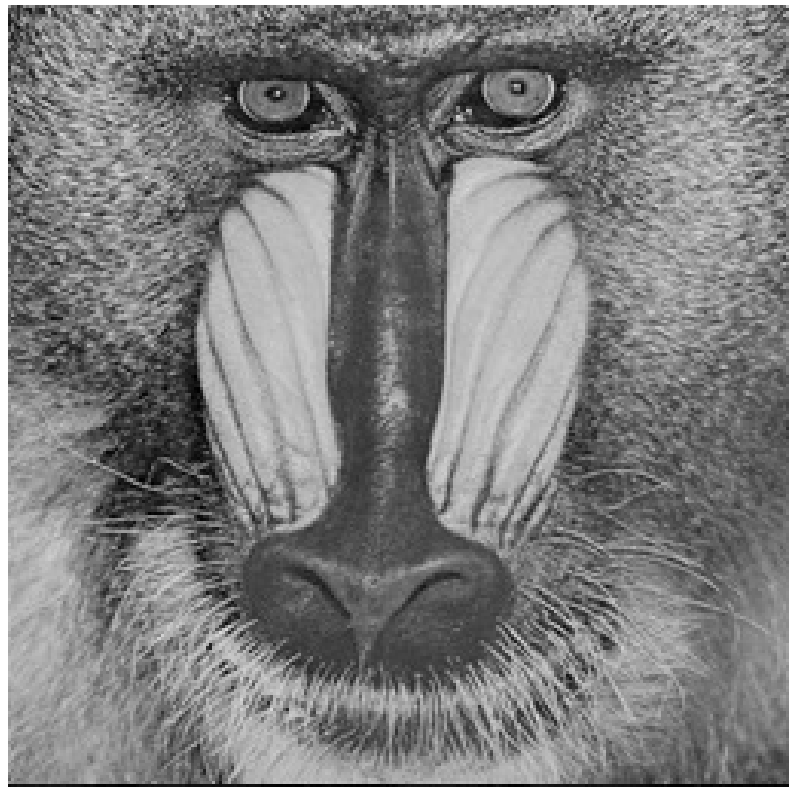
If we instead optimize a twodimensional vector quantizer for the speech signal with rate 2 bits/sample (ie $2^{2 \cdot 2} = 16$ vectors in the codebook), the decision regions and reconstruction points look like



SNR is approximately 14.4 dB.

LBG, example

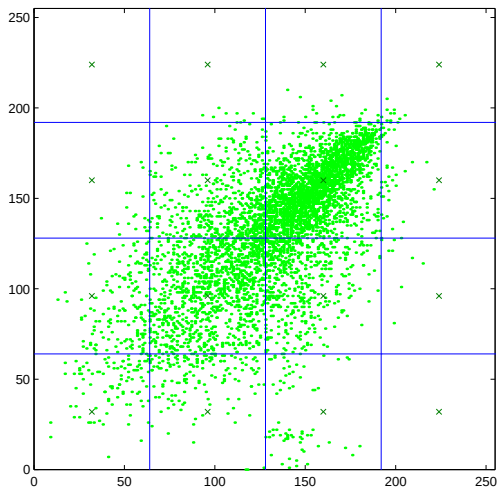
We want to code this image (256×256 pixels, 8 bits/pixel).



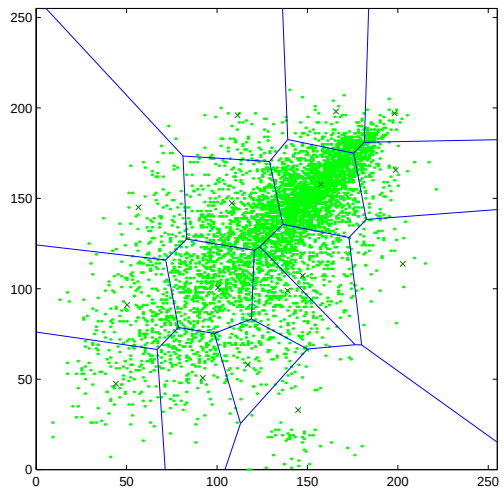
Training data: 10000 pixels viewed as 5000 2-dimensional vectors.
We want a rate of 2 bits/pixel, which means that we have
$M = 2^{RL} = 2^{2 \cdot 2} = 16$ reconstruction points.

LBG, example

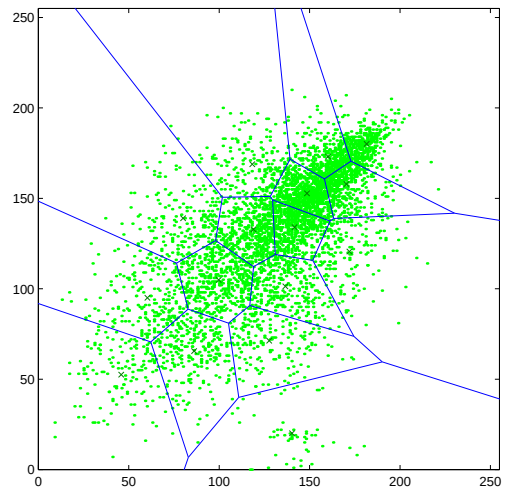
The figures show the training vectors in green, and the reconstruction points and their corresponding decision regions.



Starting codebook



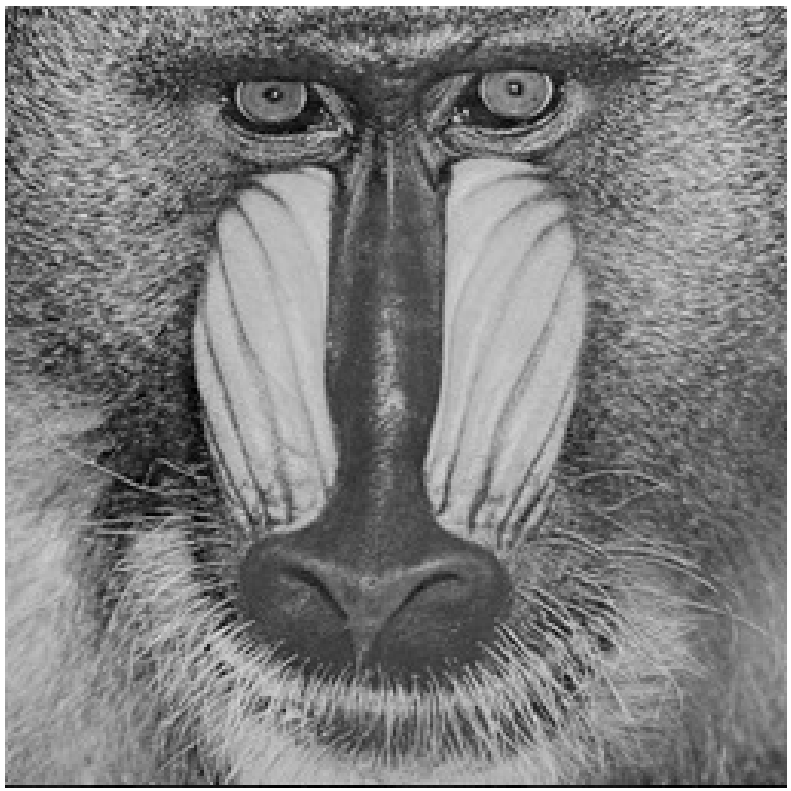
1 iteration



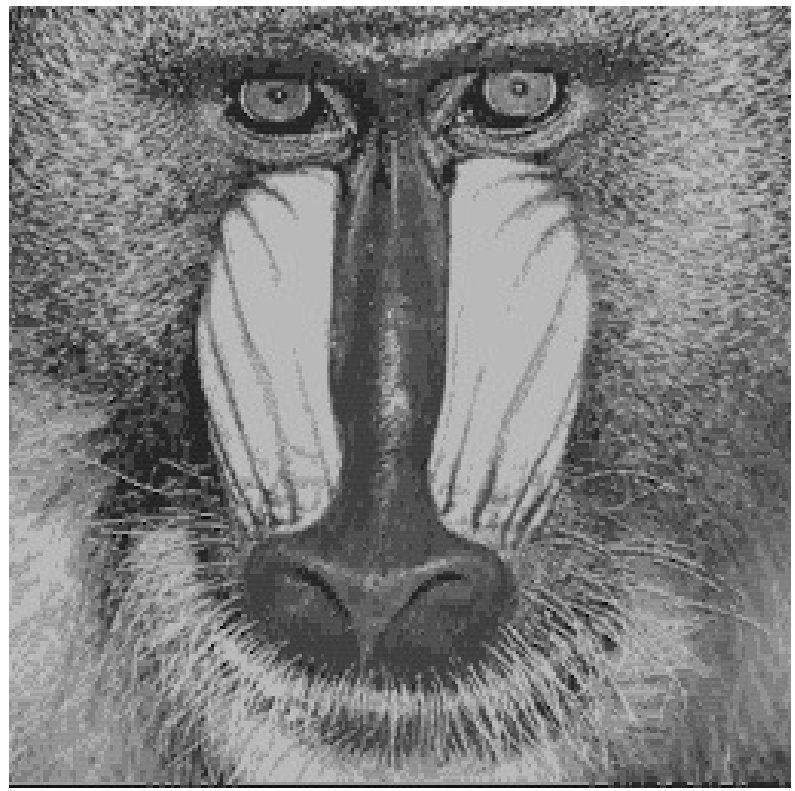
10 iterations

LBG, example

If the image is coded with the resulting quantizer we get the result



Original



Quantized

The resulting SNR is 19.25 dB.
The side information (the codebook) requires $16 \cdot 2 \cdot 8 = 256$ bits.

Tree-Structured Vector Quantization

Place the reconstruction vectors as leaves in a binary tree. In each inner node of the tree we store two test vectors.

When we want to quantize a vector, we start by comparing it to the two test vectors in the root node. Depending on which test vector that is closest to the signal we descend along that branch. Keep doing this at each new node until we reach a leaf, ie a reconstruction point.

The advantage is that we only need to perform $2 \cdot \log_2 M$ comparisons instead of M comparisons (binary search vs full search).

The disadvantage is that we might not end up with the reconstruction point that is closest, ie the distortion will be a little higher compared to a full search quantizer. The storage requirements will also be larger, since we have to store all test vectors too.

A TSVQ can be designed using a variant of the splitting algorithm.

Gain-shape VQ

A vector $\mathbf{x}$ can de described as

$$\mathbf{x} = ||\mathbf{x}|| \cdot \frac{\mathbf{x}}{||\mathbf{x}||}$$

We construct a scalar quantizer for the length ("gain") of the vector $||\mathbf{x}||$ and a vector quantizer for the normalized vector ("shape") $\frac{\mathbf{x}}{||\mathbf{x}||}$.

Lower complexity compared to a full search vector quantizer, but the distortion will be a little higher.

Multistage VQ

Instead of having a single L-dimensional quantizer with rate R bits/sample, we use k L-dimensional quantizers with rates $R_1, R_2, \ldots, R_k$, such that $R = R_1 + R_2 + \ldots + R_k$.

Start by quantizing $\mathbf{x}$ using quantizer 1. The quantization error $\mathbf{x}_1 = \mathbf{x} - Q_1(\mathbf{x})$ is then quantized using quantizer 2. The quantization error $\mathbf{x}_2 = \mathbf{x}_1 - Q_2(\mathbf{x}_1)$ is quantized using quantizer 3, and so on.

The total number of vectors over all k codebooks is $2^{LR_1} + 2^{LR_2} + \ldots + 2^{LR_k}$.
In comparison, a regular vector quantizer has a codebook of size $2^{LR} = 2^{LR_1} \cdot 2^{LR_2} \cdot \ldots \cdot 2^{LR_k}$.

We get a quantizer with lower complexity (smaller codebook and fewer comparisons) at the cost of higher distortion.

Lattice VQ

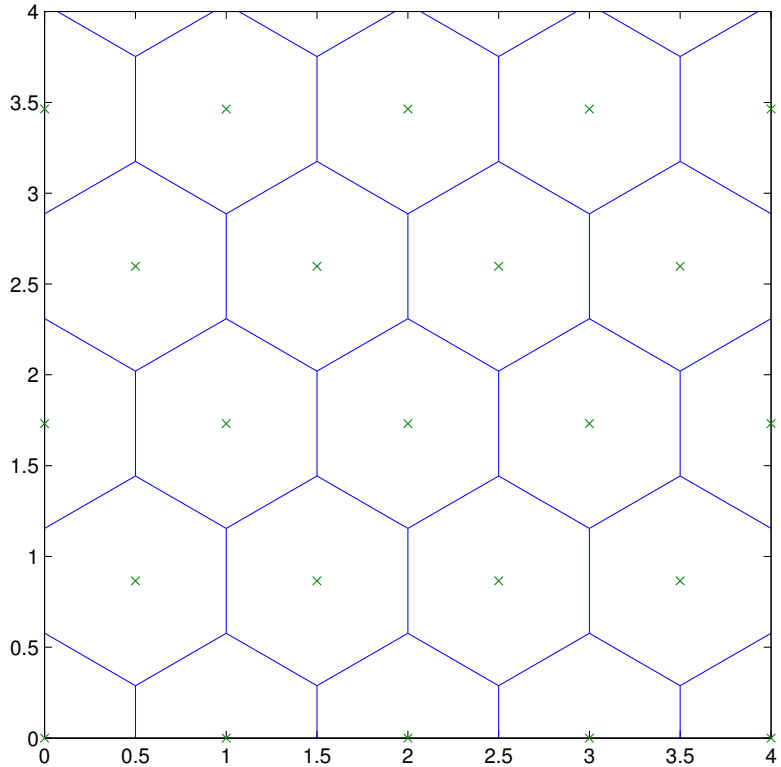
Generalization of uniform quantization to multiple dimensions. The reconstruction points are placed in a regular pattern (a lattice) in the L-dimensional space.

Usually low complexity, since we don't have to store all reconstruction points but only a description of the pattern. For most lattices there are fast methods of finding the closest reconstruction point, so there is no need to do a full search when quantizing.

Gives lower distortion for the same rate, compared to a uniform scalar quantizer.

Lattice quantization, cont.

Example of a lattice quantizer in two dimensions: Hexagonal lattice.



At fine quantization the hexagonal lattice gives a gain of approximately 0.17 dB, compared to uniform scalar quantization.
The best known lattice is a 24-dimensional lattice that gives a gain of 1.03 dB compared to uniform scalar quantization.