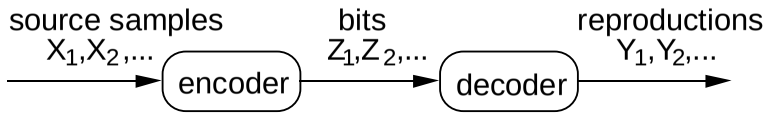
VECTOR QUANTIZATION (VQ)

Introduction

- *Source code* = *encoder* + *decoder*



We assume source samples are real-valued.

The encoder receives the sequence of source samples $X_1, X_2, \ldots$ and produces a sequence of bits $Z_1, Z_2, \ldots$, which is the *binary representation* of the source samples. The encoder is also said to *encode* the source samples, and to produce *encoded bits*.

The decoder receives the binary representation and produces a sequence of *reproductions* or, equivalently, *reconstructions* of the source samples.

- *Vector Quantizer*[1]: For fixed integers k and L, it is a source code such that
  - Encoder operates independently on successive nonoverlapping *blocks* (equivalently, *vectors*) of k successive source samples. More specifically, there is a function α, called the *encoding rule* , such that the encoder produces

$$Z_1,\ldots,Z_L = \alpha(X_1,\ldots,X_k), \quad Z_{L+1},\ldots,Z_{2L} = \alpha(X_{k+1},\ldots,X_{2k}),$$

$$Z_{2L+1},\ldots,Z_{3L} = \alpha(X_{2k+1},\ldots,X_{3k}), \quad \ldots$$

[1]Also known as a *block code*,

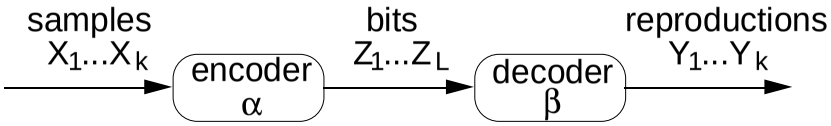
  - Decoder operates independently on successive nonoverlapping blocks of L successive bits. More specifically, there is a function β, called the *decoding rule* , such that the decoder produces

$$Y_1,\ldots,Y_k = \beta(Z_1,\ldots,Z_L), \quad Y_{k+1},\ldots,Y_{2k} = \beta(Z_{L+1},\ldots,Z_{2L}),$$

$$Y_{2k+1},\ldots,Y_{3k} = \beta(Z_{2L+1},\ldots,Z_{3L}), \quad \ldots$$

We will assume that β is a one-to-one function, i.e. β assigns distinct values to distinct arguments. We will see later that this entails no potential loss of optimality.

- For example, the encoding of the first vector of source samples is illustrated below.



- k is called the *vector dimension*, *blocklength* or *size*, *VQ dimension*
- L is called the *codelength*
- This kind of source code is also called a

  fixed-rate or fixed-length (memoryless) vector quantizer (VQ).

- This kind of source code is *lossy* because the encoding rule maps real-valued vectors into bits. As such, it cannot be an invertible, i.e. lossless, operation. Therefore, $Y_1,...,Y_k$ cannot always equal $X_1,...,X_k$.
- The sets of all possible encoder outputs and all possible decoder outputs play important roles.
- The partition induced on space k-dimensional input vectors plays an important role
- Fixed-rate VQ is a fairly general paradigm that includes many lossy source codes as special cases, e.g. fixed-rate transform coding. Since it is quite general and also analyzable, it provides an excellent framework for studying lossy source codes.
- JPEG can be viewed as a variable-rate, rather than fixed-rate, VQ. Except for the encoding of dc coefficients, it operates independently on bocks of 64 pixels.

ADDITIONAL CHARACTERISTICS OF A VQ

- The *binary codebook* C_b:

$$C_b = \{\alpha(\underline{x}): \underline{x} \in \mathcal{R}^k\}, \qquad (\underline{x} \text{ is shorthand for } (x_1,...,x_k))$$

$$= [\underline{c}_1, \underline{c}_2, \ldots, \underline{c}_M] \subset \{0,1\}^L \qquad \text{(an ordered set)}$$

where $\underline{c}_i = (c_{i,1},...,c_{i,L}) \in \{0,1\}^L$ is the ith *binary codeword*.

- M is the *size* of the VQ.

$M \leq 2^L$, since there are 2^L binary sequences of length L.

- The *reproduction codebook* or *codebook* C:

$$C = [\beta(\underline{c}_1), \beta(\underline{c}_2), ..., \beta(\underline{c}_M)] \qquad \text{(an ordered set)}$$

$$= [\underline{w}_1, \underline{w}_2, \ldots, \underline{w}_M] \subset R^k$$

where $\underline{w}_i = (w_{i,1},\ldots,w_{i,k}) \in R^k$ is the ith *codevector*

Codevectors are also called *codepoints*, or *reproduction/reconstruction vectors/ points.*

Note the convention:

$$\beta(\underline{c}_i) = \underline{w}_i$$

- The *quantization rule* or *reproduction rule* Q:

$$Q(\underline{x}) = \beta(\alpha(\underline{x}))$$

This summarizes the encoding and decoding from $\underline{X}$ to $\underline{Y}$. But it loses track of the bits produced by the encoder.

- The *quantizing/encoding partititon* S:

$$S = [S_1, S_2, \ldots, S_M] \qquad \text{(an ordered set)}$$

where M is called the *size* of the partition and the VQ, and

$$S_i = \{\underline{x} \in R^k: \alpha(\underline{x}) = \underline{c}_i\} = \{\underline{x} \in R^k: Q(\underline{x}) = \underline{w}_i\}$$

is the ith *quantizing /encoding cell*

Definition: A *partition* of R^k is a collection of disjoint subsets of R^k whose union is R^k. The elements of the collection are called *cells*.

It follows that the encoding rule is

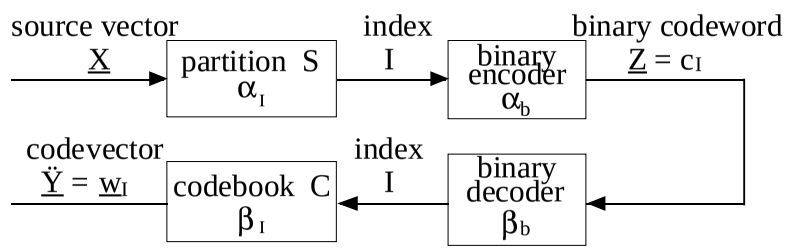
$$\alpha(\underline{x}) = \underline{c}_i \text{ when } \underline{x} \in S_i$$

and the quantization rule is

$$Q(\underline{x}) = \underline{w}_i \text{ when } \underline{x} \in S_i$$

MORE DETAILED BLOCK DIAGRAM OF A VQ

It is useful to decompose the encoder and decoder into two blocks:



Encoder: The first block *implements* the partition: It finds the quantizing cell in which $\underline{x}$ lies, and outputs its *index* I, i.e. $I = \alpha_I(\underline{X})$, where

$$\alpha_I(\underline{x}) = i \text{ if } \underline{x} \in S_i$$

The second block is the *binary encoder*, which assigns the index its binary codeword, i.e. $\underline{Z} = \alpha_b(I)$, where

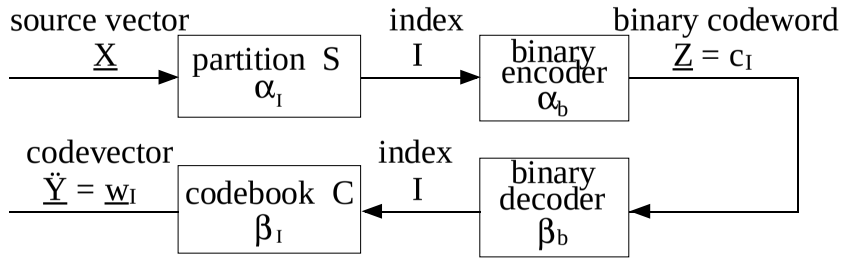
$$\alpha_b(i) = \underline{c}_i$$

α_I is called the *index encoding rule* or *partition rule*[2]

α_b is called the *binary encoding rule*. It is one-to-one.

$\alpha(\underline{x}) = \alpha_b(\alpha_I(\underline{x}))$

[2]These are not standard names. Better suggestions would be appreciated.



Decoder: The first block is the *binary decoder*. It inverts the binary encoder, recreating the index I, i.e. $I = \beta_b(\underline{Z})$, where

$$\beta_b(\underline{z}) = i \text{ when } \underline{z} = \underline{c}_i$$

The second block produces the codevector indexed by I, i.e. $\underline{Y} = \beta_I(I)$,

$$\beta_I(i) = \underline{w}_i$$

β_β is called the *binary decoding rule*

β_I is called the *codebook rule*[2]

$\beta(\underline{z}) = \beta_I(\beta_b(\underline{z}))$

SUMMARY: A VQ IS CHARACTERIZED BY

Dimension: k

Codeword length: L

Size: M

Encoding rule: α

characterized by

Partition: $S = [S_1, S_2, \ldots, S_M]$

Binary codebook: $C_b = [\underline{c}_1, \underline{c}_2, \ldots, \underline{c}_M]$

Encoding rules:

$$\alpha(\underline{x}) = \underline{c}_i \text{ when } \underline{x} \in S_i, \quad \alpha_I(\underline{x}) = i \text{ when } \underline{x} \in S_i, \quad \alpha_b(i) = \underline{c}_i$$

Decoding rule: β

characterized by

Codebook: $C = [\underline{w}_1, \underline{w}_2, \ldots, \underline{w}_M]$

Decoding rules: $\beta(\underline{z}) = \underline{w}_i$ when $\underline{z} = \underline{c}_i$, $\beta_b(\underline{z}) = i$ when $\underline{z} = \underline{c}_i$, $\beta_I(i) = \underline{w}_i$

Quantization rule: Q (determined by S and C)

$$Q(\underline{x}) = \beta(\alpha(\underline{x})) = \underline{w}_i \text{ when } \underline{x} \in S_i$$

Most important characteristics (all others can be deduced from these)

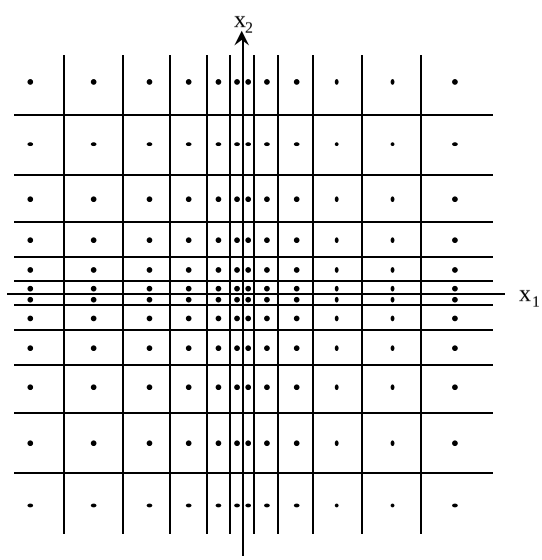
k, L, M, S, C

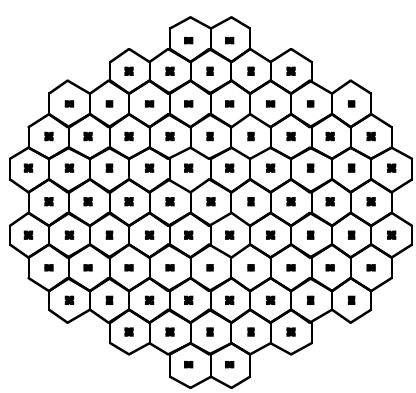
Notes

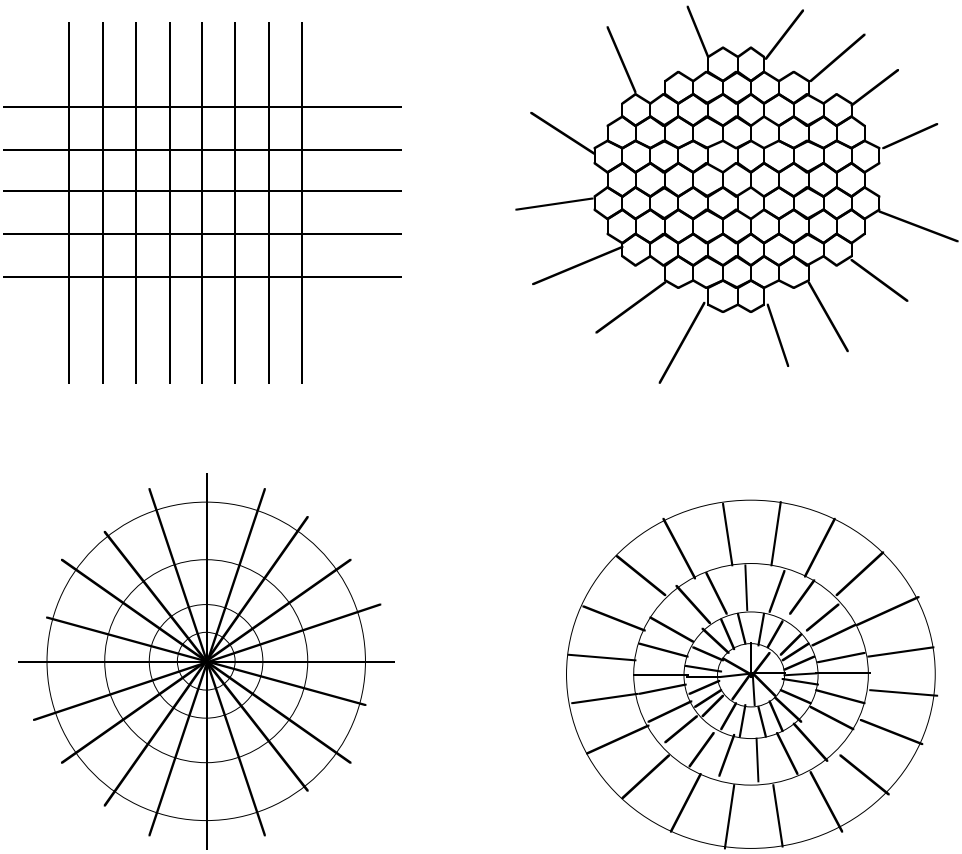
- When thinking about VQ, many people think first about the codebook C.
- It is usually better to think first about the partition S.
- You must learn to use terminology and notation on the previous pages.

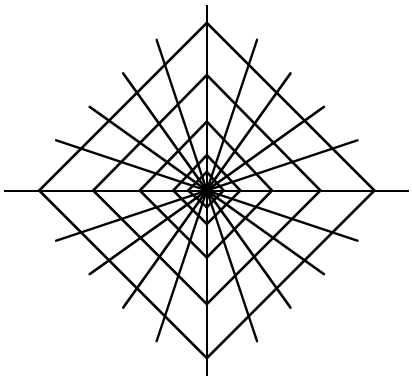
EXAMPLES

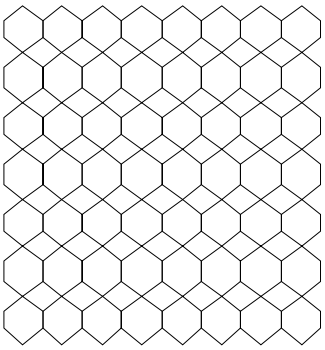
(in k=2 dimensions)

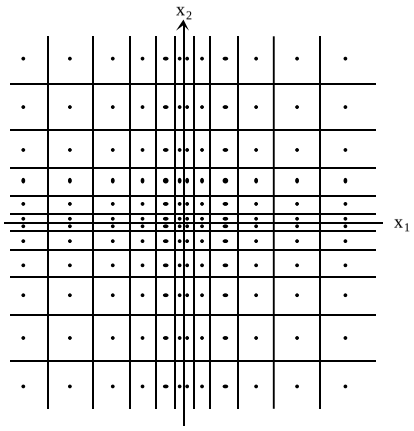


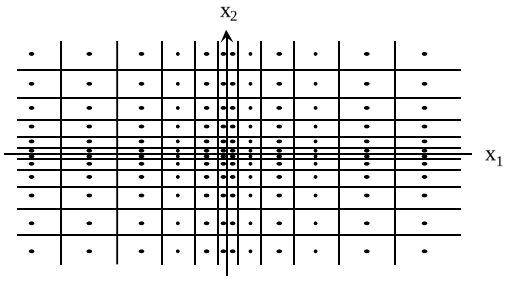


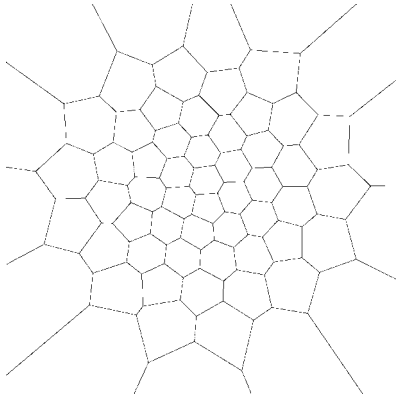


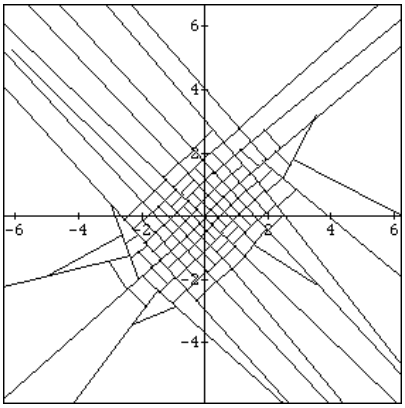


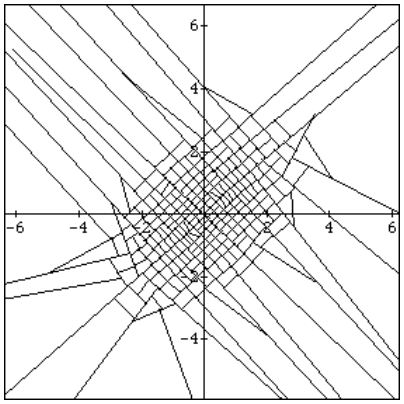



x_2
x_1

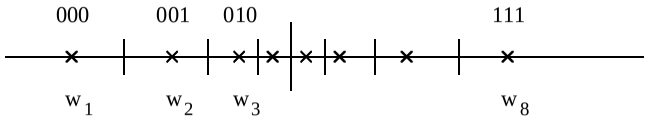

x_2
x_1







Scalar Quantizer (k=1)



PERFORMANCE

Performance is measured by two quantities: *encoding rate* and *distortion*.

ENCODING RATE

Encoding rate is a measure of the encoding efficiency. Roughly speaking, it is the number of encoded bits produced per source sample. More specifically for a fixed-rate VQ it is:

$$R = \frac{L}{k} \text{ bits/sample}$$

One of our principal goals is to have the rate R be small.

Sometimes we want an expression for rate in terms of k and M, instead of k and L. Suppose we are given the dimension k and the size M, which is the number of quantization cells, or equivalently the number of possible cell indices. Then we must choose L large enough that the number of binary sequences of length L is at least as large as M. That is, we must have $2^L \geq M$, or equivalently, $L \geq \log_2 M$. Indeed, the smallest possible value of L is $\lceil \log_2 M \rceil$, which leads to the smallest possible rate, namely,

$$R = \frac{\lceil \log_2 M \rceil}{k}$$

If M is not a power of two, we can reduce the rate somewhat if we allow the binary encoder to encode m successive indices simultaneously, for some integer m. In this case, L must be large enough that the number of binary sequences of length L is at least as large as M^m, the number of possible sequences of m quantization indices. That is, we must have $2^L \geq M^m$, or equivalently, $L \geq \log_2 M^m = m \log_2 M$. Indeed, the smallest possible value of L is $\lceil m \log_2 M \rceil$, which leads to the smallest possible rate, namely,

$$R = \frac{\lceil m \log_2 M \rceil}{mk}$$

where we have divided L by mk because this is the number of source samples that are being encoded by $\lceil m \log_2 M \rceil$ bits. If m is large, then

$$R \cong \frac{\log_2 M}{k}.$$

Thus, from now on, unless there is need to be picky, we will take the definition of the rate of a size M dimension k fixed-rate VQ to be

$$R = \frac{\log_2 M}{k} \text{ bits/sample}$$

with the understanding that to actually code at approximately this rate, we may have to simultaneously binary encode blocks of successive quantization indices.

Note: Rate is entirely determined by the encoder, not the decoder.

DISTORTION

Distortion is a measure of the accuracy of $\underline{Y}$ produced by decoder as a reproduction of the original source samples in $\underline{X}$. In this course, we will primarily use *mean-square error* (MSE) as the measure of distortion.

We assume that $\underline{X} = (X_1,\ldots,X_k)$ is a jointly continuous random vector whose probability density is denoted $f_{\underline{X}}(\underline{x})$. The mean-squared error is given by the notation and formulas below:

$$D = D_{\underline{X}}(Q) = D_{\underline{X}}(S,C) \qquad \text{(the subscript } \underline{x} \text{ will often be omitted)}$$

$$= \frac{1}{k}\sum_{i=1}^{k} E(X_i-Y_i)^2 = \frac{1}{k} E \sum_{i=1}^{k} (X_i-Y_i)^2$$

$$= \frac{1}{k} E\,||\underline{X}-\underline{Y}||^2 = \frac{1}{k} E\,||\underline{X}-Q(\underline{X})||^2 = \frac{1}{k}\int ||\underline{x}-Q(\underline{x})||^2 f_{\underline{X}}(\underline{x})\, d\underline{x}$$

$$= \frac{1}{k}\sum_{i=1}^{M} \int_{S_i} ||\underline{x}-\underline{w}_i||^2 f_{\underline{X}}(\underline{x})\, d\underline{x}$$

where

$$||\underline{x}-\underline{y}|| = \sqrt{\sum_{i=1}^{k}(x_i-y_i)^2} = \text{Euclidean distance between } \underline{x} \text{ and } \underline{y}$$

and where expected value is with respect to probability distribution on $\underline{X} = (X_1,\ldots,X_k)$.

Notes:

1. The above is actually the definition of *statistical average* distortion and MSE. It is used when we have a probabilistic model for the source vector. When we actually run the VQ on real data, we measure *empirical average* distortion and MSE

   $$D_{emp} = \frac{1}{n}\sum_{i=1}^{n}(X_i-Y_i)^2$$

   where $n >> k$ is the length of the data.

   If the data comes from a stationary, ergodic source, then when n is large

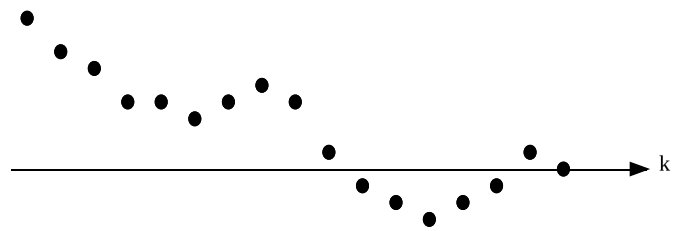
   $$D_{emp} \cong D$$

2. Distortion is entirely determined by Q, or equivalently the partition S and codebook C. The binary encoding and decoding rules do not affect the distortion.

EXAMPLE OF SCALAR AND VECTOR QUANTIZATION

Source sequence x_k:

This could be the output of a highly correlated source.



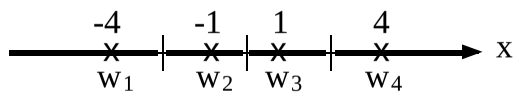
A scalar quantizer:

k=1, M=4

$C_1 = \{w_1, w_2, w_3, w_4\} = \{-4, -1, 1, 4\}$

= codebook of *quantization levels*

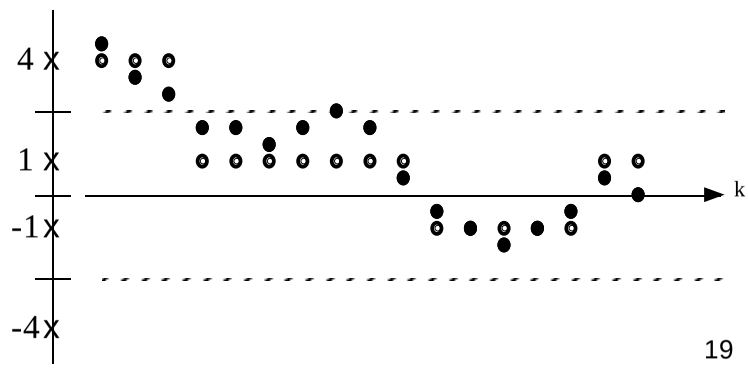
levels and cells shown →



R = 2 bits/samp

The scalar quantizer applied to the source sequence:

The open circles show the quantized outputs

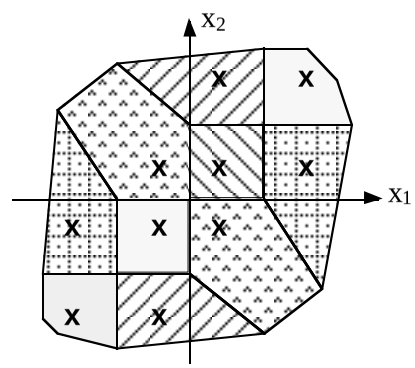


A Vector Quantizer:

k = 2, M = 10, $R = \frac{1}{2}\log_2 10 = 1.66$ bits/sample

$C_2 = \{ (-4,-4), (-4,-1), (-1,-4), (-1,-1), (-1,1), (1,-1), (1,1), (1,4), (4,1), (4,4) \}$

The codevectors and quantization cells →



The idea behind this VQ:

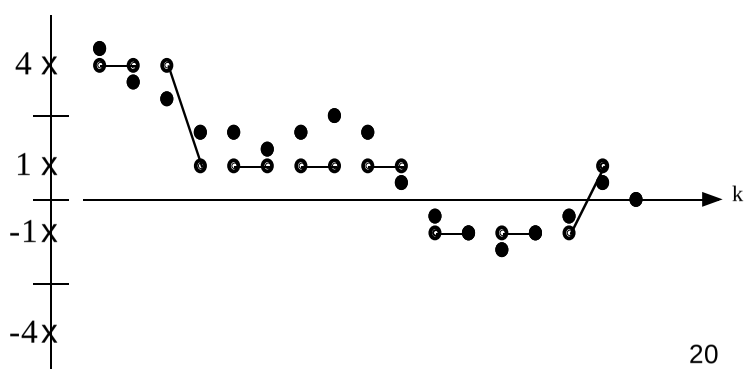
When the scalar quantizer is applied to a slowly changing data sequence, successive source samples are usually quantized into identical or adjacent "levels". The codebook C_2 consists of just those pairs having identical or adjacent components. In comparison, the scalar quantizer permits any of the 16 possible pairs of successive levels in the set

$$C_1 \times C_1 = \{ (-4,-4), (-4,-1), (-4,1), (-4,4), (-1,-4), \ldots, (4,4) \}.$$

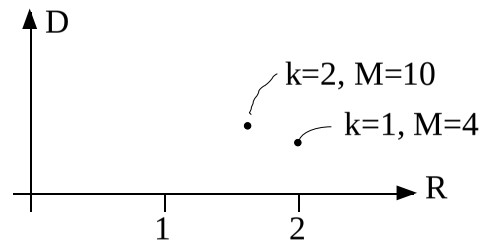
Since 10 < 16, the VQ has lower rate than the scalar quantizer.

The result of the VQ applied to the source sequence is shown →

Lines connecting open circles indicate chosen codevectors.



The distortion and rate for the scalar and vector quantizers:



(The distortions shown are not actual MSE, just representative values.)

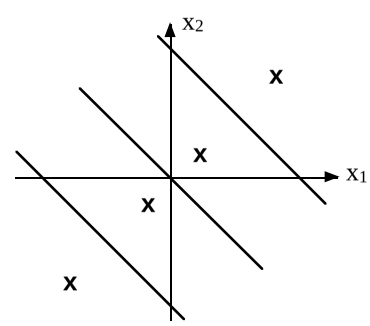
Another vector quantizer:

k = 2

M = 4

C_3 = { (-4,-4), (-1,-1), (1,1), (4,4) }

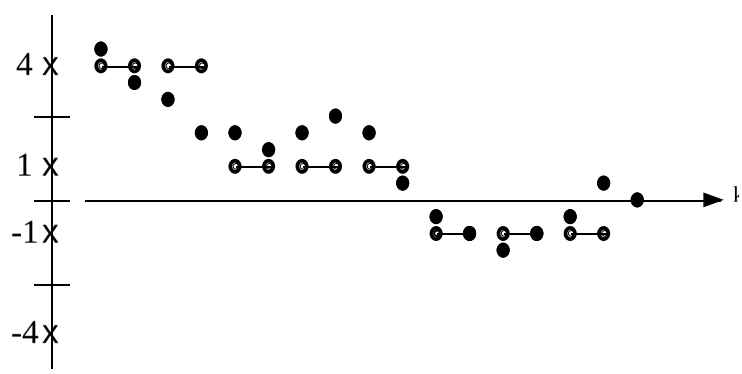
The codevectors and quantization cells →



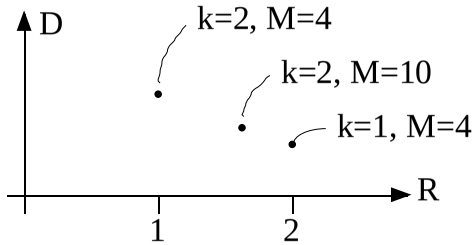
$R = \frac{1}{2}\log_2 4 = 1$ bit/sample

This VQ applied to the source sequence is shown →

Lines connecting open circles indicate chosen codevectors.



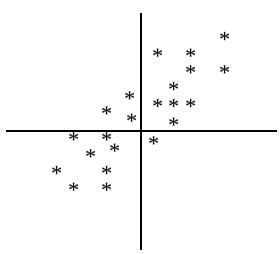
This quantizer has even less rate, and somewhat larger distortion, as illustrated below.



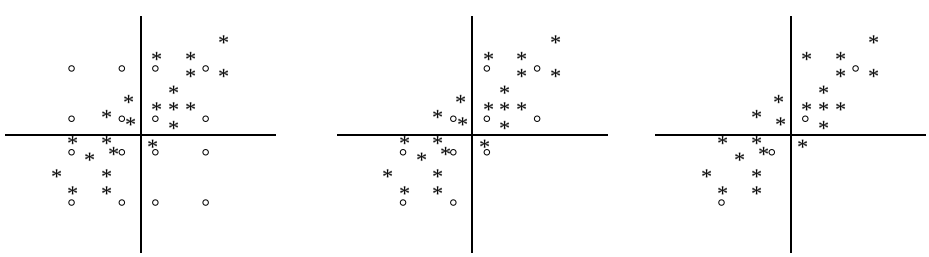
(again, the distortions are not actual, just representative)

ANOTHER VIEW

Scatter plot of typical (x_1,x_2) pairs



scalar quant. used twice VQ with M = 10 VQ with M = 4



OVERLOAD AND GRANULAR REGIONS AND DISTORTION

For a typical quantizer, one may subdivide k-dimensional space roughly into two regions:

- The granular region R_G, consisting of all $\underline{x}$ such that $||\underline{x}-Q(\underline{x})||^2 \lesssim \alpha D$, where $\alpha \cong 3$
- The overload region R_o, consisting of all $\underline{x}$ such that $||\underline{x}-Q(\underline{x})||^2 > \alpha D$

There is no widely accepted precise definition of overload and granular regions, but roughly speaking granular region is where the quantization errors are small, i.e. the "noise" added by quantization is "granular", and the overload region is where distortion is large, i.e. where the quantizer is said to be "overloaded".

The choice of $\alpha \cong 3$ above is motivated by uniform scalar quantization with step size Δ for which it is natural to define the overload region to begin $\Delta/2$ beyond the extreme levels. This is where $||\underline{x}-Q(\underline{x})||^2 > (\Delta/2)^2$, which is three times larger than the common approximation $D \cong \Delta^2/12$.

It is often useful to decompose the distortion into granular and overload distortions. That is,

$$D = D_G + D_O$$

where

$$D_G = \text{granular distortion} = \frac{1}{k}\int_{R_G} ||\underline{x}-Q(\underline{x})||^2 f_{\underline{X}}(\underline{x})\, d\underline{x}$$

$$D_O = \text{overload distortion} = \frac{1}{k}\int_{R_O} ||\underline{x}-Q(\underline{x})||^2 f_{\underline{X}}(\underline{x})\, d\underline{x}$$

Typically, when a quantizer is designed to have small distortion, $D_o << D_G$

KEY QUESTIONS:

- How to implement the encoder, i.e. the partitioning?
- Complexity?
- How to design/optimize fixed-rate VQ's?

  (What properties do good fixed-rate VQ's have?)
- How to estimate MSE of a VQ?
- How to design low complexity VQ's with good performance?
- What is best possible performance (D vs. R) of a VQ? (the opta function) How does it depend on dimension k?
- How well do low complexity VQ's perform? What is it in their structure that limits their performance?

OUTLINE OF VQ COVERAGE

- Optimality properties of fixed-rate VQ's.
- "Full search" encoding.
- Generalized-Lloyd iterative VQ design algorithms
- Properties of optimal quantizers,

  e.g. $E\,||\underline{Y}||^2 = E\,||\underline{X}||^2 - E\,||\underline{X}-\underline{Y}||^2$
- High-Resolution Analysis of MSE --

  Bennett's integral for VQ
- High-resolution analysis of optimal performance -- Zador-Gersho formula
- Comparison to Shannon's rate-distortion theory analysis of optimal performance
- Variable-rate VQ -- optimality properties and high-resolution theory.

VQ REFERENCES

1. A. Gersho & R. Gray, *Vector Quantization and Signal Compression*, Klewer. Excellent reference. Very full coverage of VQ, except for high resolution theory.
2. R. Gray, "Vector quantization," *IEEE ASSP Magazine*, April 1984. A good brief introduction to VQ.
3. R. Gray and D.L. Neuhoff, "Quantization," *IEEE Trans. Inform. Theory*, Oct. 1998. A concise introduction to VQ. This paper has a good summary of the history, theory and practice of VQ. It has the only summary of high-resolution theory. It also has an extensive set of references.

QUANTIZATION THEORY

There are two generic domains in which quantization theory can proceed, both analysis and design.

In the *random vector domain*, the input data to be quantized is a k-dimensional random vector $\underline{X} = (X_1,...,X_k)$. We operate in this domain when the dimension k is specified in advance. For example, in CELP speech coders there are typically 10 or so AR coefficients to be quantized. In this domain, we can use k-dimensional VQ, or we can divide $\underline{X}$ into subvectors and encode each with a lower dimensional VQ.

In the *random process domain,* the data is a random process $X_1, X_2, X_3, \ldots$. For example, the data might be an infinite sequence of speech samples. In this domain, we are free to choose any k as the dimension of VQ. In fact, we can even use k_1-dimensional VQ to encode the first k_1 samples, k_2-dimensional VQ to encode the next k_2 samples, and so on.

In either domain, we will assume that the random variables $X_1, X_2, \ldots$ are jointly continuous random variables, whose distribution is described by a joint probability density function (pdf) $f_{\underline{X}}(\underline{x})$.

As suggested perhaps by the discussion on p. 1 of these notes, we will mostly be interested in the random process domain. However, once k is fixed, we are effectively in the random vector domain.

Finally, when in the random process domain, unless otherwise stated, we will assume that the random process is stationary. In this way, we can consider the effects of changing the VQ dimension k, knowing that any change in performance is due to the change in dimension, rather than the fact that different kinds of random variables are being encoded.

OPTIMAL QUANTIZERS

Definition:

A quantizer (S,C) or Q with dimension k and size M is said to be *optimal for* $\underline{X}$, *k , M*, if it has the smallest MSE of any quantizer with dimension k and size M applied to the random vector $\underline{X}$.

Facts:

- For any $\underline{X}$, k, M, there always exists at least one optimal quantizer.

  (The proof involves the fact that any continuous functions on a closed and bounded set has a minimum.)
- In some cases, there is more than one optimal quantizer.

Definitions:

$\delta_{\underline{X}}(k,M)$ = least MSE of any quantizer with size M for $\underline{X} = (X_1,\ldots,X_k)$.

$\delta_{\underline{X}}(k,R)$ = least MSE of any quantizer with rate R for $\underline{X} = (X_1,\ldots,X_k)$.

These are called OPTA[3] functions (Optimum Performance Theoretically Attainable)

[3]It's not a very nice acronym. For one thing the word "theoretically" is redundant. Suggestions for a better acronym would be welcomed.

When operating in the random process domain we consider the data to be encoded to be a random process $X_1, X_2, \ldots$ which we denote simply by X. In this case the OPTA functions will be denoted $\delta_X(k,M)$ and $\delta_X(k,R)$, respectively.

PARTIAL OPTIMALITY CONDITIONS

There are two important partial optimality conditions. The first specifies the optimal partition for a given codebook. The second does the reverse.

Optimal partition for a given codebook

Suppose we are given a codebook $C = \{\underline{w}_1,\ldots,\underline{w}_M\}$ and a random vector $\underline{X}$, and suppose we seek the best partition $S = \{S_1,\ldots,S_M\}$ to use with this codebook. That is, we seek the partition S that minimizes the distortion $D(S,C)$. To see how to choose S, consider the MSE

$$D(S,C) = \frac{1}{k}\int \|\underline{x}-Q(\underline{x})\|^2 f_{\underline{X}}(\underline{x})\, d\underline{x}$$

where $Q(\underline{x}) = \underline{w}_i$ when $\underline{x} \in S_i$. Thus, choosing S is tantamount to choosing Q.

From the above we see that to make D small, one cannot do better than, for each $\underline{x}$, to choose $Q(\underline{x})$ to be a codevector $\underline{w}_i$ such that $\|\underline{x}-\underline{w}_i\| \le \|\underline{x}-\underline{w}_j\|$ for all $j \ne i$. This means that if $\underline{x}$ is closer to $\underline{w}_i$ than to any other $\underline{w}_j$, then $\underline{x}$ should be placed in cell S_i. In other words, the cell S_i should contain the set

$$V_i = \left\{ \underline{x} : \|\underline{x}-\underline{w}_i\| < \|\underline{x}-\underline{w}_j\|, \text{ for all } j \ne i \right\}.$$

If $\underline{x}$ is equally closest to two or more of the codevectors, then we can place $\underline{x}$ in the cell of any one of the closest codevectors.

On the other hand, the set W of all $\underline{x}$'s that are equally closest to two or more codevectors is a (k-1)-dimensional subset of k-dimensional space, and since $\underline{X}$ has a jointly continuous distribution, this set has zero probability. Since the integral expression for the distortion D is not influenced by the value of $Q(\underline{x})$ on a set of probability zero, it doesn't actually matter how these $\underline{x}$'s are placed in cells. The distortion will be the same no matter what. Indeed, distortion will not be influenced, even if the S_i's contain all of the points closest to $\underline{w}_i$ except for a set of probability zero. This discussion is summarized in the following:

Optimality Condition 1: Optimal partition(s) for a given codebook.

Given a codebook $C = \{\underline{w}_1,\ldots,\underline{w}_M\}$, let

$$V_i = \left\{\underline{x} : \|\underline{x}-\underline{w}_i\| < \|\underline{x}-\underline{w}_j\|, \text{ for all } j \ne i\right\}.$$

A partition $S = \{S_1,\ldots,S_M\}$ minimizes MSE for the given codebook and random source vector $\underline{X}$ if and only if

$$S_i \doteq V_i \text{ for each } i,$$

where $A \doteq B$ means $\Pr\left(\underline{X} \in (A-B) \cup (B-A)\right) = 0$.

Notes:

- It is interesting to notice that an optimal partition can be chosen without regard to the probability distribution of $\underline{X}$.
- One may view that the role of the encoder is to "control" the decoder to produce the best possible output. That is, the encoder should output a binary codeword that causes the decoder to output a codevector that is closest to the source vector $\underline{x}$. This is precisely what an optimal partition for a given codebook causes the encoder to do. This reinforces why the optimal partition need not depend on the probability distribution of $\underline{X}$. Specifically, if the source produces $\underline{x}$, the decoder should produce $\underline{y}$ as close as possible to $\underline{x}$ and "how close" should not depend on $f_{\underline{X}}(\underline{x})$.
- A partition S such that $S_i \supseteq V_i$ for all i is called a "Voronoi partition"; its cells are called "Voronoi cells". Other names for this partition are "nearest neighbor", "Dirichlet".

  Voronoi partitions are unique except for the points that are not contained in any of the V_i's. That is,

  $$S_i = V_i \cup T_i$$

  where T_i is some subset (possibly empty) of the points that are closest to $\underline{w}_i$ as well to some other point. That is, T_i is a subset of

  $$\{\underline{x} : ||\underline{x}-\underline{w}_i|| = ||\underline{x}-\underline{w}_j|| \text{ some } j, \text{ and } ||\underline{x}-\underline{w}_i|| \le ||\underline{x}-\underline{w}_j|| \text{ for all } j\}$$

  All T_i's have probability zero.

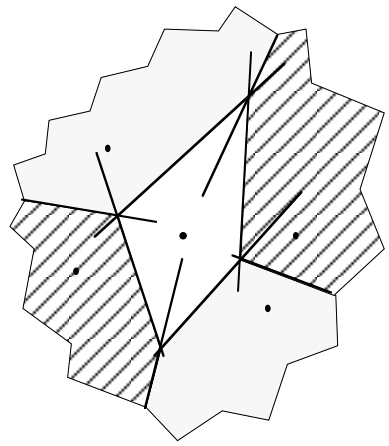
- Ordinarily, we won't fuss about sets of probability zero, and about how the points that are equidistant between codevectors are assigned to codevectors. We'll usually simply say that "the optimal partition or the Voronoi partition is"

  $$S_i = \{\underline{x} : ||\underline{x}-\underline{w}_i|| < ||\underline{x}-\underline{w}_j||, \text{ for all } j \neq i\}$$

  or

  $$S_i = \{\underline{x} : ||\underline{x}-\underline{w}_i|| \le ||\underline{x}-\underline{w}_j||, \text{ for all } j \neq i\}.$$

- To find the Voronoi partition, draw perpendicular bisectors between each pair of codevectors. These are (k-1)-dimen'l hyperplanes, each dividing R^k into two half spaces. The Voronoi region S_i is the intersection of the halfspaces containing $\underline{w}_i$.
- Voronoi cells are convex polyhedra, i.e. the intersection of a set of halfspaces.
- 2-Dimensional Example:



- Exercise: Show that if three points are not colinear, their three perpendicular bisectors meet at a point.

<u>Optimal codebook for a given partition</u>

Suppose we are given a partition $S = \{S_1,\ldots,S_M\}$ and a random vector $\underline{X}$, and suppose we seek the best codebook $C = \{\underline{w}_1,\ldots,\underline{w}_M\}$ to use with this partition. That is, we seek the codebook C that minimizes the distortion $D_{\underline{X}}(S,C)$. The answer comes immediately when one considers that the decoder is, in effect, told which quantization cell S_i the source vector $\underline{X}$ resides and must produce a value $\underline{Y}$ that minimizes $E\,||\underline{X}-\hat{\underline{X}}||^2$. This value $\underline{Y}$ may be considered to be an estimate of $\underline{X}$. In this light, the role of the decoder is to estimate $\underline{X}$ from knowledge of the cell in which $\underline{X}$ resides. From conventional minimum MSE estimation theory, we know that when the decoder is told $\underline{X}\in S_i$, it should produce the conditional expectation $E[\underline{X}|\underline{X}\in S_i]$. That is, the optimal codevector for the cell S_i is $E[\underline{X}|\underline{X}\in S_i]$.

A direct argument, which in effect rehashes the derivation of the minimum MSE estimation rule, follows:

$$D_{\underline{X}}(S,C) = \sum_{i=1}^{M} \Pr(\underline{X}\in S_i) \int_{S_i} ||\underline{x}-\underline{w}_i||^2 \frac{f_{\underline{X}}(\underline{x})}{\Pr(\underline{X}\in S_i)}\, d\underline{x} = \sum_{i=1}^{M} \Pr(\underline{X}\in S_i) \int ||\underline{x}-\underline{w}_i||^2 f_{\underline{X}}(\underline{x}|\underline{X}\in S_i)\, d\underline{x}$$

$$= \sum_{i=1}^{M} \Pr(\underline{X}\in S_i)\, E[||\underline{X}-\underline{w}_i||^2|\underline{X}\in S_i]$$

The sum is minimized by minimizing each term, i.e. by choosing $\underline{w}_i = E[\underline{X}|\underline{X}\in S_i]$. (Recall: $E||\underline{X}-\underline{w}||^2$ is minimized by $\underline{w} = E[\underline{X}]$.)

This discussion is summarized in the following:

<u>Optimality Condition 2:</u> Optimal codebook for a given partition.

Given a partition $S = \{S_1, \ldots, S_M\}$ and source density $f_{\underline{X}}(\underline{x})$, the unique codevectors that minimize MSE are the "centroids"

$$\underline{w}_i = E[\underline{X}|\underline{X}\in S_i] = \int \underline{x}\, f_{\underline{X}}(\underline{x}|\underline{X}\in S_i)\, d\underline{x} \;, \quad i = 1,\ldots,M$$

where

$$f_{\underline{X}}(\underline{x}|\underline{X}\in S_i) = \begin{cases} \dfrac{f_{\underline{X}}(\underline{x})}{\Pr(\underline{X}\in S_i)}, & \underline{x}\in S_i \\ 0\;, & \text{else} \end{cases}$$

Notes:

- There is one and only one optimal codebook for a given partition.
- Unlike the optimal partition for a given codebook, the optimal codebook for a given partition does indeed depend on the the source prob. distribution.

We now combine the two Optimality Conditionsl

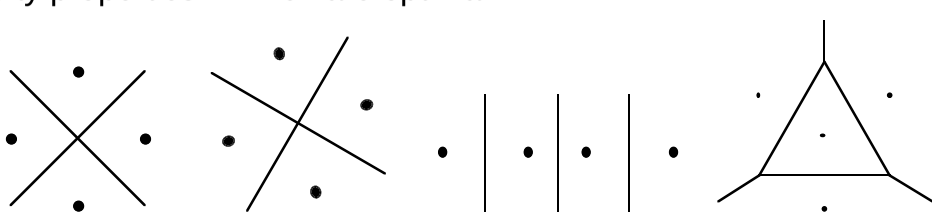
<u>Corollary:</u> Optimal partition and codebook.

Any optimal fixed-rate VQ (i.e. one with smallest MSE for the given k, M and $\underline{X}$) has partition satisfying (*) and codebook satisfying (**).

<u>Notes:</u>

- There may be more than one quantizer that satisfies both optimality conditions, even when there is only one optimal quantizer. In this case, at least one of them must be optimal.

  Example: k =2, M=4, (X_1,X_2) is IID Gaussian. All of the following satisfy both optimality properties. Which are optimal?



- If there is only one quantizer that satisfies the optimality criteria, it is the one and only optimal quantizer.

- Let $k=1$. If the logarithm of the source density $f(x)$ is strictly convex $\cap$, then it can be shown that there exists one and only one quantizer that satifies the optimality criteria and consequently, one and only one optimal quantizer. (Fleischer, 1964)
- Example: The log of a Gaussian density (zero mean and unit variance) is $-x^2/2 - \log\sqrt{2\pi}$, which is strictly convex $\cap$. Therefore, there is only one quantizer that satisfies both optimality conditions, and this is the optimal quantizer.
- Example: The log of a Laplacian density (zero mean and unit variance) is $-\sqrt{2}|x| - \log\sqrt{2}$, which is convex $\cap$, but not strictly convex $\cap$. Therefore, the condition stated above does not apply. Nevertheless, it is known that for this case there is only one quantizer that satisfies both optimality conditions, and this is the optimal quantizer. (Fleischer 1964)
- For $k \geq 2$, I know of no similar condition for checking the uniqueness of quantizers satisfying the optimality criteria. Indeed, there is not likely to be one, because as illustrated by the multiple quantizers that satisfy the optimality conditions for a pair of IID Gaussian variables, there can be a number of quantizers that satisfy both optimality criteria even when the joint density is very ordinary .

THE OPTIMALITY CONDITIONS INSURE LOCAL OPTIMALITY

Fact: A quantizer is locally optimal iff it satisfies (*) and (**)

Defn: A VQ is *locally optimal* if all sufficiently small perturbations increase or maintain distortion.

What is meant by "sufficiently small perturbation"?

Replace a codevector $\underline{w}_i$ by $\underline{w}_i + \varepsilon \underline{z}$ where $\underline{z}$ is an arbitrary vector $\underline{z}$ and ε is some scalar. If the VQ is locally optimal, then for any $\underline{z}$ there is an ε_o such that for all $\varepsilon \leq \varepsilon_o$, the perturbed VQ has the same or larger distortion than the original VQ. Any number of codevectors can be perturbed. Alternatively, consider moving or stretching the boundary of some cell by an amount "proportional" to ε. Then there must exist some ε_o such that for all $\varepsilon \leq \varepsilon_o$, the perturbed VQ has the same or larger distortion.

A VQ is locally optimal if for all possible perturbations (of any number of codevectors and any number of ways of changing boundaries), there is an ε_o such that for all $\varepsilon \leq \varepsilon_o$, the perturbed VQ has the same or large distortion.

Sketch of Proof of Fact: Local opt $\Rightarrow$ (*) and (**): If a quantizer does not satisfy (*) and (**), then it can be improved by small perturbations, so it is not locally optimal. The contrapositive of this is: Locally opt $\Rightarrow$ (*) and (**).

(*) and (**) $\Rightarrow$ local opt: If a quantizer satisfies (*) and (**), then it is locally optimal because any small perturbation will cause it not to satisfy the (*) and (**), and in either case the MSE will increase.

BRUTE FORCE IMPLEMENTATION OF "UNSTRUCTURED" VQ

One of the questions posed earlier, was: How to implement a VQ? Of particular the question of how to implement the partition rule. The following is the usual method of implementing the optimal partition S for a given codebook C.

Full-Search Encoding

Store the codebook $C = \{\underline{w}_1,\ldots,\underline{w}_M\}$.

1. Given $\underline{x}$ compute $||\underline{x}-\underline{w}_i||^2$ for each i
2. Find the i that minimizes $||\underline{x}-\underline{w}_i||^2$ and send $\underline{c}_i$

Table-Lookup Decoding

Store the codebook $C = \{\underline{w}_1,\ldots,\underline{w}_M\}$ in a table..

When the decoder is given $\underline{Z} = \underline{c}_i$, it outputs $\underline{Y} = \underline{w}_i$ as the reproduction of $\underline{X}$.

Notes:

- This is the basic form of "unstructured" VQ. It is unstructured because, in comparison to methods presented later, neither the encoder nor decoder exploits any "structure" in the partition or codebook.
- When people speak of ordinary VQ, this is often what they mean.

COMPLEXITY OF UNSTRUCTURED VQ

Storage: codebook must be stored at encoder and decoder

	storage	ops/sample
encoder:	Mkb	3M
decoder:	Mkb	0

b = no. of bits/component $\cong$ R + 3 to 5 is usually sufficient

Encoding operations:

M distance squared's, each requiring k subtracts,
k squarings, (k-1) adds, M-1 comparisons

Summary: complexity is proportional to M

The "curse of dimensionality"

Since $M=2^{kR}$, storage and computation increase exponentially with k and R.

The dimension-rate product kR is the determining factor. For example, k=64 and R=1, then $M=2^{64} = 1.8\times10^{19}$.

This is a critical limitation. Generally, in practice

$kR \leq 10$ or 12.

While modern computers can implement somewhat larger codebooks, designing them can be extremely difficult, as will be discussed later.

There are also:

- Fast search methods for unstructured VQ codebooks.

  We will discuss some later.

  Though they can be much simpler, I've yet to see one whose complexity does not increase exponentially with the distortion-rate product, except in the scalar case.
- For scalar quantizers, a simple binary search can find the closest codevector in $\log_2 M = kR$ steps. Storage can also be reduced.
- Structured VQ methods

  For example, JPEG

  We will discuss others later.

  Their codebooks and/or partitions are structured in a way that permits fast encoding.

  Their partition might not be Voronoi.

  Their codevectors might not be centroids.

  Their performance might not be as good as an "optimal" VQ with the same dimension and rate, but their lower complexity might permit a larger dimension and better performance for the same complexity.

GENERALIZED LLOYD VQ DESIGN ALGORITHMS

These are iterative algorithms in the spirit of Lloyd's algorithm[4] for scalar quantizer design. Given k, M and $p(\underline{x})$, they seek a locally optimum fixed-rate vector quantizer by alternating between

1. finding the best partition for the most recently found codebook, and
2. finding the best codebook for the most recently found partition.

They stop when 1 and 2 make no change, or an insignificant change, in which case (*) and (**) are nearly satisfied; i.e. a locally optimum VQ is approached. A block diagram of such an algorithm is shown below

Issues:

1. Initial codebook
2. Basic iterative steps
3. Stopping conditions
4. Convergence of the algorithm

[4]Stephen Lloyd, unpublished Bell Labs memorandum July 1957. Much later, a version was published as: "Least squares quantization in PCM," *IEEE Trans. Inform. Theory*, vol. IT-28, p. 129-137, Mar. 1982.

1. Initial codebook:

There are many possible choices. Some common ones:

(a) A set of M representative source sequences. They might be generated from the pdf with a random number generator.

(b) The k-fold Cartesian product of the codebook of optimal scalar quantizer with $M^{1/k}$ levels.

(c) The set formed by adding an additional codevector in close proximity to each codevector of an optimal k-dimensional VQ with $M/2$ codevectors. In this method, one starts by designing a VQ with 2 codevectors, then successively designs VQ's with twice the size of the previous one.

2. Basic iterative steps

Given $C^{(i)} = \{\underline{w}_1^{(i)},\ldots,\underline{w}_M^{(i)}\}$ produced by the ith iteration,

(a) $S^{(i+1)} = \{S_1^{(i+1)}, \ldots, S_M^{(i+1)}\}$, where

$$S_j^{(i+1)} = \{\underline{x} : \|\underline{x}-\underline{w}_j^{(i)}\| < \|\underline{x}-\underline{w}_{j'}^{(i)}\|,\ j'\neq j\},\quad j = 1,\ldots,M$$

(b) $C^{(i+1)} = \{\underline{w}_1^{(i+1)}, \ldots, \underline{w}_M^{(i+1)}\}$, where

$$\underline{w}_j^{(i+1)} = E[\underline{X}|\underline{X}\in S_j^{(i+1)}],\quad j = 1,\ldots,M$$

Since storing or describing the partition found in (a) can be difficult, these two steps are usually combined into the following single step:

(c) $$\underline{w}_j^{(i+1)} = E\left[\underline{X}\,\middle|\,\|\underline{x}-\underline{w}_j^{(i)}\| < \|\underline{x}-\underline{w}_{j'}^{(i)}\|,\ j'\neq j)\right]$$

3. Stopping conditions

Possibilities include:

(a) Stop when codevectors change little, i.e. when

$$||W_j^{(i+1)} - w_j^{(i)}|| < \varepsilon, \quad j = 1,\ldots,M,$$

where ε is some small tolerance.

(b) Stop when distortion changes little, i.e. when

$$|D(S^{(i+1)},C^{(i+1)}) - D(S^{(i)},C^{(i)})| < \varepsilon$$

where ε is some small tolerance.

The ε's in the above are usually chosen to be much, much smaller than the distortion of the quantizer.

4. Convergence issues

Each step of the algorithm maintains or decreases the training distortion.

(a) $D(S^{(i+1)},C^{(i)}) \leq D(S^{(i)},C^{(i)})$

(b) $D(S^{(i+1)},C^{(i+1)}) \leq D(S^{(i+1)},C^{(i)})$

Therefore,

$$D(S^{(i+1)},C^{(i+1)}) \leq D(S^{(i)},C^{(i)})$$

The sequence of distortions $D(S^{(1)},C^{(1)})$, $D(S^{(2)},C^{(2)})$, $D(S^{(3)},C^{(3)})$, ... is nonincreasing and nonnegative. Therefore, from a basic theorem of analysis, it has a limit. Thus we can say that the distortions of the VQ's designed by this algorithm converge to a limit.

Do the codebooks and partition also converge to limits? In practice, the answer is generally, yes. But I don't know what convergence properties have been proven. It is conceivable that the algorithm might reach a *limit cycle*. For example, it might be that after some number of iterations it designs quantizer (S,C), then (S',C'), then (S,C), and so on. (If this happens, both will quantizers have the same same distortion.)

Because of the potential for locally optimal quantizers that are not globally optimal, it is wise to run the algorithm with several different initial codebooks.

DESIGNING VQ'S FROM A TRAINING SEQUENCE

The generalized Lloyd algorithm is seldom used in the form described so far, is sometimes used for designing scalar quantizers, but it is seldom used for designing vector quantizers ($k \geq 2$) because

- the pdf of the source vector is often difficult to quantify.
- working with a k-dimensional pdf's (e.g. computing the k-dimensional integrals involved in centroid calculations) is prohibitively complex.

Therefore, in practice VQ's are almost always designed by "training sequence" methods such as the algorithm described next. This algorithm is often called the LBG algorithm, because it was first introduced in a paper by Linde, Buzo and Gray[5]. The algorithm was also introduced in the pattern recognition literature for a somewhat different purpose as the "k-means algorithm".

[5] J. Linde, A. Buzo, R. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. 28, pp. 84-95, Jan. 1980.

THE LBG ALGORITHM

We wish to design a k-dimensional VQ with size M

Training Sequence: This algorithm presumes we have a representative sequence of k-dimensional vectors from the source, $\underline{t}_1, \underline{t}_2, \ldots, \underline{t}_N$, called a *training sequence* of *training vectors*. If the source density is known, the training sequence could be generated by a random number generator. More commonly, one takes a long sample from the source to be quantized.

LBG Algorithm:

- Choose an initial codebook $C^{(0)} = \{\underline{w}_1^{(0)}, \ldots, \underline{w}_M^{(0)}\}$.
- Iterate the following steps until the centroids change little and/or the distortion changes little.

  Given $\tilde{C}^{(i)} = \{\tilde{\underline{w}}_1^{(i)}, \ldots, \tilde{\underline{w}}_M^{(i)}\}$ produced by the ith iteration,

  1. Partition training set: $\tilde{S}^{(i+1)} = \{\tilde{S}_1^{(i+1)}, \ldots, \tilde{S}_M^{(i+1)}\}$

     $$\tilde{S}_j^{(i+1)} = \{\, \underline{t}_n : \underline{t}_n \text{ closer to } \tilde{\underline{w}}_j^{(i)} \text{ than to } \tilde{\underline{w}}_{j'}^{(i)},\ j' \neq j \,\}$$

  2. Find empirical centroids:

     $$\tilde{\underline{w}}_j^{(i+1)} = \frac{1}{N_j^{(i+1)}} \sum_{n:\, \underline{t}_n \in \tilde{S}_j^{(i+1)}} \underline{t}_n ,$$

     where $N_j^{(i+1)}$ = # training vectors in $\tilde{S}_j^{(i+1)}$.

In Step 1, for each training vector $\underline{t}_n$, we apply full search encoding to find the the closest codevector in $\tilde{C}^{(i)}$. The training vector and the index of the closest codevector are stored in a table (an $N \times 2$ array).

Step 2 is implemented by counting and averaging all training vectors with a given label.

Alternatively, one may combine these two steps into the following single step:

3. Given $\tilde{C}^{(i)} = \{\underline{\tilde{w}}_1^{(i)},\ldots,\underline{\tilde{w}}_M^{(i)}\}$ produced by the ith iteration,

$$\underline{\tilde{w}}_j^{(i+1)} = \frac{1}{N_j^{(i)}} \sum_{n:\, \underline{t}_n \text{ closest to } \underline{\tilde{w}}_j^{(i)}} \underline{t}_n ,$$

where $N_j^{(i)}$ = # training vectors closest to $\underline{\tilde{w}}_j^{(i)}$

In other words, for each training vector $\underline{t}_n$, one finds the find codevector, say $\underline{\tilde{w}}_j^{(i)}$, to which it is closest, via full search. One increments a counter that stores $N_j^{(i)}$, and one adds $\underline{t}_n$ to an accumator that computes the sum for $\underline{\tilde{w}}_j^{(i)}$.

Issues:

1. Initial codebook: Same options as before.
2. Basic iteration: Already discussed.
3. Stopping conditons: Same options as before.
4. Convergence:

   As before, each step maintains or decreases the *training distortion* :

$$D_{train} = \frac{1}{N} \sum_{n=1}^{N} ||\underline{t}_n - Q(\underline{t}_n)||^2$$

   Therefore, the training distortion converges to some value. Usually, the quantizer converges to a local optimum, i.e. to a quantizer that satisfies the empirical versions of optimality conditions (1) and (2).

   Moreover, since there are only a finite number of distinct partitions of the given training sequence, after a finite number of steps the algorithm necessarily reaches a local optimum after which it does not change, or it reaches a limit cycle, i.e. it cycles repeatedly through some finite number of partition-codebook pairs. However, the algorithm is usually stopped long before either of these occur.

   Because the algorithm works with a finite set of training vectors, there tend to be more local optima than with algorithms that work directly with the pdf such as the first generalized Lloyd algorithm. Therefore, it is usually wise to rerun the algorithm with several different choices of initial codebook.

5. Software

We have a version of the LBG algorithm available in C.
Other versions can be found on the web.

6. Alternate derivation of the LBG algorithm:

Instead of hypothesizing an unknown pdf , suppose that we know only that we have a training sequence $\underline{t}_1, \underline{t}_2, \ldots, \underline{t}_N$, and that we desire to choose codevectors $\underline{w}_1,\ldots,\underline{w}_M$ to minimize the training distortion

$$D_{train} = \sum_{n=1}^{N} \min_j \|\underline{t}_n - \underline{w}_j\|^2$$

To attempt to do this, repeatedly perform the following:

$$\tilde{\underline{w}}_j^{(i+1)} = \frac{1}{N_j^{(i)}} \sum_{n:\, \underline{t}_n \text{ closest to } \tilde{\underline{w}}_j^{(i)}} \underline{t}_n ,$$

where $N_j^{(i)}$ = # training vectors closest to $\tilde{\underline{w}}_j^{(i)}$
Performing the above will not increase and usually decrease

$$\sum_{n=1}^{N} \min_j \|\underline{t}_n - \underline{w}_j\|^2$$

7. Training distortion vs. actual distortion:

Suppose that the LBG algorithm is run on a training sequence $\underline{t}_1, \underline{t}_2, \ldots, \underline{t}_N$ taken from a source with probability density $p(\underline{x})$, and it designs a quantizer Q Then, the *actual distortion*

$$D = E(\underline{X}-Q(\underline{X}))^2$$

of Q will usually be greater than the training distortion,

$$D_{train} = \frac{1}{N} \sum_{n=1}^{N} \|\underline{t}_n - Q(\underline{t}_n)\|^2$$

which is the distortion measured on the training sequence.

To see that actual and training distortion can be different, consider the extreme case where the training sequence length N equals the size M of the desired VQ. In this case, the algorithm will choose the codebook to be the training sequence itself, and it will find the training distortion to be zero, whereas the actual distortion will definitely be larger. Moreover, the designed quantizer is probably far from optimal.

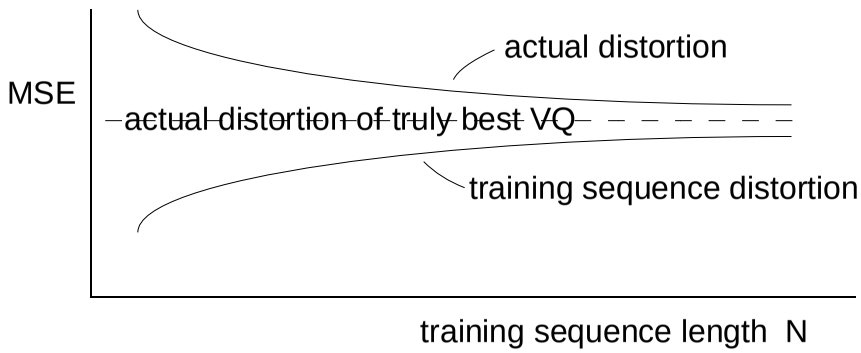
As illustrated on the next page, the difference between training and actual distortions usually becomes smaller as N increases.

Because training distortion can be substantially less than actual distortion, it is customary to estimate the actual distortion by running the VQ on a *test sequence* that is distinct from the training sequence.

Training sequence length:

How large to make the training sequence length N?

Note: As illustrated below, the VQ designed by the algorithm usually becomes better as N is made larger, and the training distortion becomes a more accurate estimate of actual distortion.



A typical rule of thumb is that N should be at least 50 M, and larger is better. Larger N is needed if training distortion is to be used as an estimate of actual distortion. (Actually, the size of N should also increase with dimension k.)

Another strategy, which is fairly conservative, is to choose N large enough that the training and test sequence distortions are reasonably close. In this case, we can be pretty sure we have both a good quantizer and a good estimate of actual distortion.

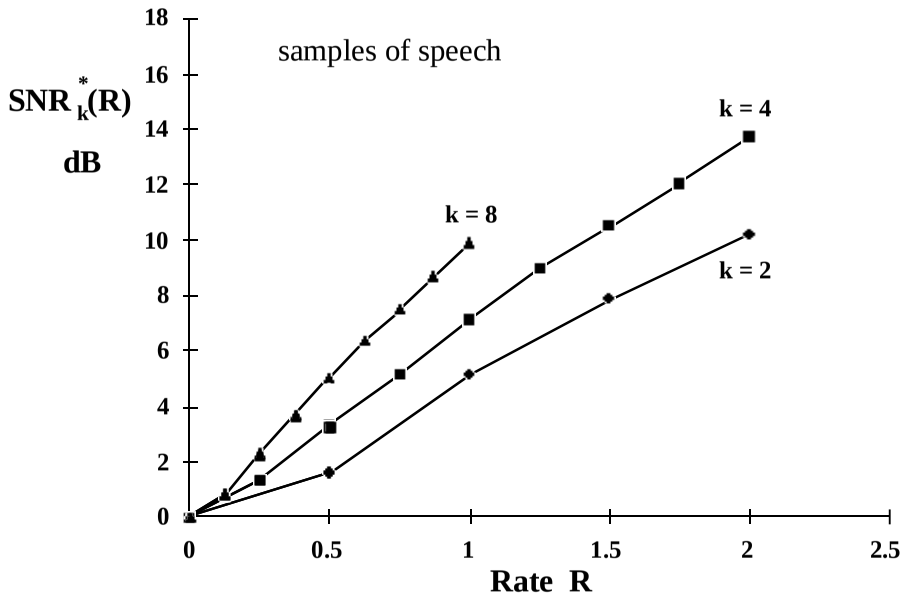
8. Complexity of VQ design

The LBG design algorithm involves encoding the training sequence a number of times and, additionally, computing the new centroids. Hence, the number of operations performed is approximately proportional to NM times the number of iterations.

Since a VQ is ordinarily designed once and used many, many times, we are willing to live with very complex VQ design algorithms. Nevertheless, design becomes a genuine problem for VQ's with moderate to large dimension-rate products. Even finding a long enough training sequence is a problem. I can't recall having seen the LBG algorithm to design an unstructured VQ with $kR > 14$. Thus, there are dimension-rate products, e.g. $kR = 20$, for which one could conceivably implement a VQ, but for which design is infeasible.

Not surprisingly, a number of reduced complexity design algorithms have been proposed. These typically involve a fast encoding algorithm, such as those mentioned previously. Another way to speed the algorithm is to make a good choice of the initial codebook. This reduces the number of iterations required.

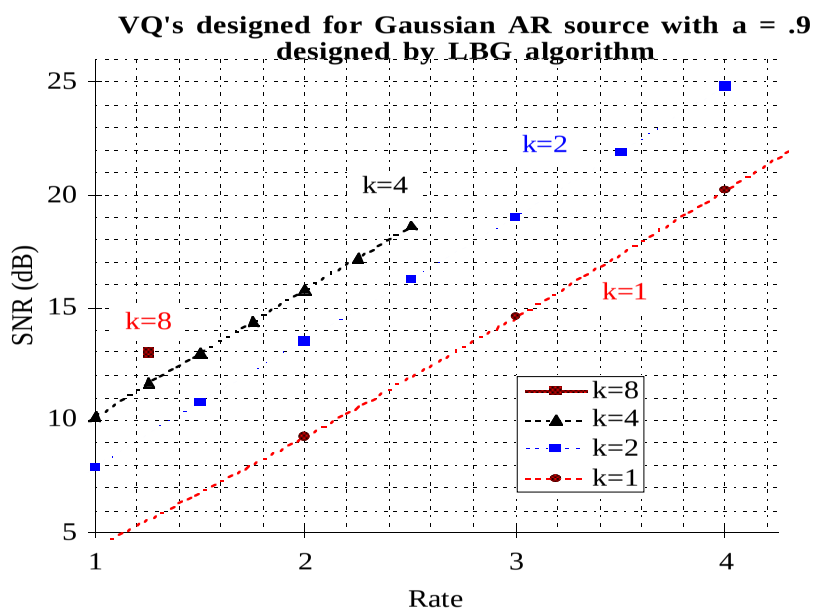
EXAMPLES OF VQ'S DESIGNED BY THE LBG ALGORITHM



k-dimensional VQ's designed using the LBG algorithm on a training sequence of 640,000 samples of speech taken at 6.5 kHz sampling rate.

Notice the gains with increasing dimension.

Notice also that for k=8, the quantizers were designed only to rate 1. This is because of the large complexity of designing higher rate VQ's with dimension 8.



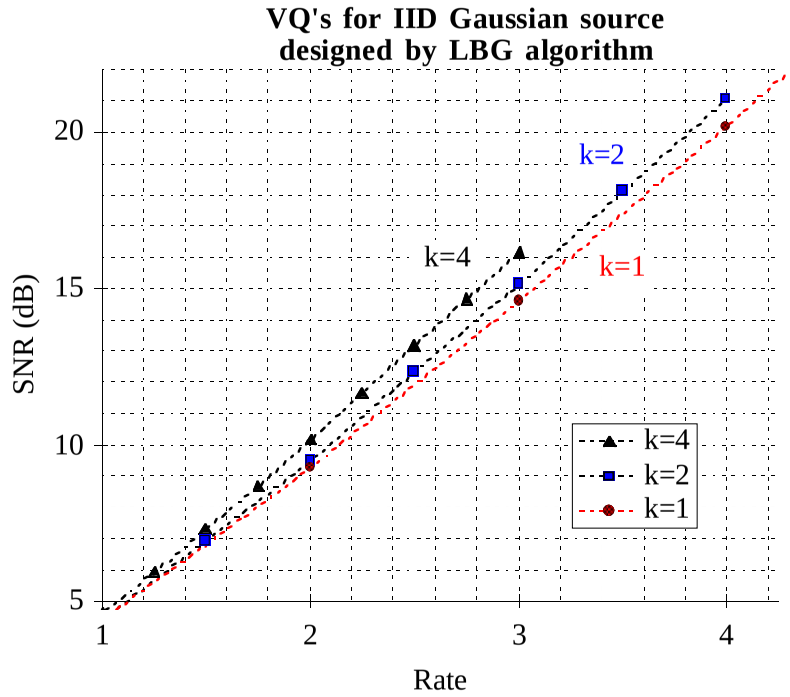
Gains (in dB) of VQ over scalar Q:

k =	2	4	8
gain ≅	4.2	6.5	7.8

Important questions to be answered by high-resolution theory:

Why does VQ gain over scalar Q? Why does gain increase with dimension?

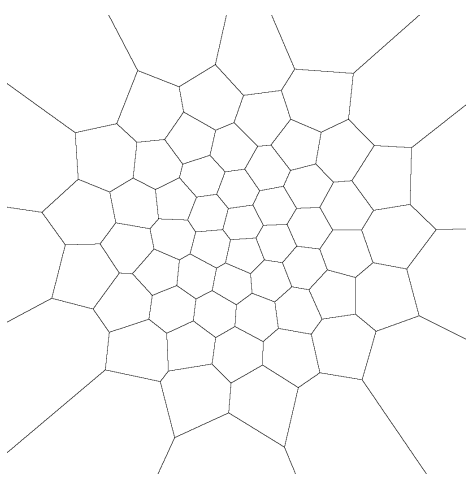
How to predict the sizes of gains? Why are these lines straight?



Again notice the gains of VQ over scalar Q (at rate 3, k=4, VQ gains 1.6 dB)
Is it intuitive that VQ gains over scalar Q, even for an IID source?
High-resolution theory will show why.

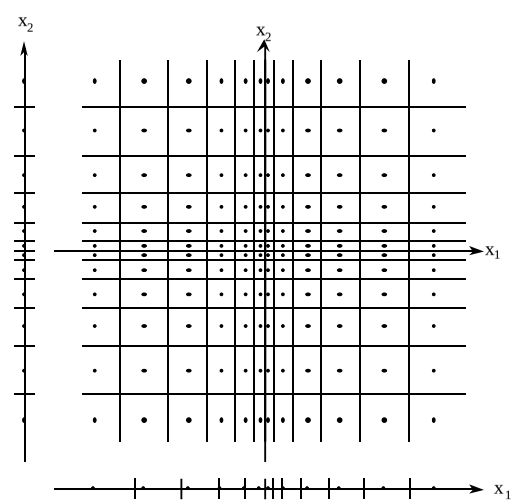
COMPARISON OF QUANTIZERS FOR AN IID GAUSSIAN SOURCE

An LBG designed 2-dimensional VQ



M = 64, R = 3

An optimal scalar quantizer used twice



M_{scalar} = 12, M = 144, R = 7.17

How are the partitions and codebooks different?

HISTORICAL NOTE

It was not until almost 1980 that vector quantization was seriously proposed and the LBG algorithm was developed. Before that, "experts" generally assumed that large dimensions would be needed for VQ's to produce significant gains over scalar quantization, probably due to their experience with channel codes, which typically have large block lengths, e.g. 100. Because they could see the large complexities associated with even moderately large dimensions, they didn't think VQ's would be useful. In fact, as the previous examples illustrate, VQ's with relatively small dimensions make sizable gains over scalar quantization. We will use high-resolution analysis to come to an understanding of these gains.

The LBG algorithm is also used in pattern recognition for identifying clusters of multidimensional features. It was developed independently in that community as the "k-means algorithm".

OTHER PROPERTIES OF OPTIMAL QUANTIZERS

Consider a VQ with

$$C = \{\underline{w}_1,\ldots,\underline{w}_M\},\quad S = \{S_1,\ldots,S_M\},\quad P_i = \Pr(\underline{X}\in S_i),\quad \underline{Y} = Q(\underline{X}) = \text{column vector.}$$

Fact A: If C satisfies the centroid property, i.e. $\underline{w}_i = E[\underline{X}|\underline{X}\in S_i]$ for all i, then

1. $E\,\underline{Y} = E\,\underline{X}$
2. $E\,X_m Y_n = E\,Y_m Y_n$
3. $E\,\underline{X}^t\underline{Y} = E\,||\underline{Y}||^2$
4. $E\,Y_m(X_n-Y_n) = 0$
5. $E\,\underline{Y}^t(\underline{X}-\underline{Y}) = 0$
6. $E\,||\underline{Y}||^2 = E\,||\underline{X}||^2 - E\,||\underline{X}-\underline{Y}||^2$
7. The centroid of a convex cell is contained in the cell.

Fact B: If S is optimal for a codebook C, then the cells of S are convex.

Facts A and B imply that locally optimal quantizers are *regular* , meaning that their cells are convex and their codevectors are contained in their cells.

Definition: A set G is *convex*, if $\alpha\,\underline{x} + (1-\alpha)\,\underline{y} \in G$ whenever $\underline{x},\underline{y} \in G$ and $0\le \alpha \le 1$.

Proof of Fact A:

Assume C satisfies centroid property, i.e. $\underline{w}_i = E[\underline{X}|\underline{X}\in S_i]$. Let $P_i=Pr(\underline{X}\in S_i)$.

1. $E\,\underline{Y} = E\,\underline{X}$

   Pf: $E\,\underline{Y} = \sum_{i=1}^{M} P_i\, E[\underline{Y}|\underline{X}\in S_i] = \sum_{i=1}^{M} P_i\, \underline{w}_i = \sum_{i=1}^{M} P_i\, E[\underline{X}|\underline{X}\in S_i] = E\,\underline{X}$

2. $E\,X_m Y_n = E\,Y_m Y_n$

   Pf: $E\,X_m Y_n = \sum_{i=1}^{M} Pr(\underline{X}\in S_i)\, E[X_m Y_n|\underline{X}\in S_i]$

   $= \sum_{i=1}^{k} Pr(\underline{X}\in S_i)\, E[X_m w_{i,n}|\underline{X}\in S_i] = \sum_{i=1}^{M} Pr(\underline{X}\in S_i)\, w_{i,n}\, w_{i,m}$

   $E\,Y_m Y_n = \sum_{i=1}^{M} Pr(\underline{X}\in S_i)\, E[Y_m Y_n|\underline{X}\in S_i]$

   $= \sum_{i=1}^{M} Pr(\underline{X}\in S_i)\, w_{i,n}\, w_{i,m} = E\,X_m Y_n$

3. $E\,\underline{X}^t\underline{Y} = E\,||\underline{Y}||^2$

   Pf: $E\,\underline{X}^t\underline{Y} = \sum_{i=1}^{k} E\,X_i Y_i = \sum_{i=1}^{k} E\,Y_i Y_i = E\sum_{i=1}^{k} Y_i^2 = E\,||\underline{Y}||^2$

   (by 2)

4. $E\,Y_m(X_n-Y_n) = 0$

   Pf: $E\,Y_m(X_n-Y_n) = E\,Y_m X_n - E\,Y_m Y_n = E\,Y_m Y_n - E\,Y_m Y_n = 0$

   (by 2)

5. $E\,\underline{Y}^t(\underline{X}-\underline{Y}) = 0$

   Pf: $E\,\underline{Y}^t(\underline{X}-\underline{Y}) = E\,\underline{Y}^t\underline{X} - E\,\underline{Y}^t\underline{Y} = E\,||\underline{Y}||^2 - E\,||\underline{Y}||^2 = 0$

   (by 3)

6. $E\,||\underline{Y}||^2 = E\,||\underline{X}||^2 - E\,||\underline{X}-\underline{Y}||^2$

   Pf: $E\,||\underline{X}-\underline{Y}||^2 = E\,||\underline{X}||^2 - 2\,E\,\underline{X}^t\underline{Y} + E\,||\underline{Y}||^2$

   $= E\,||\underline{X}||^2 - 2\,E\,||\underline{Y}||^2 + E\,||\underline{Y}||^2 = E\,||\underline{X}||^2 - E\,||\underline{Y}||^2$

   (by 3)

7. The centroid of a convex cell is contained in the cell.

   Follows from the definition of a convex cell.

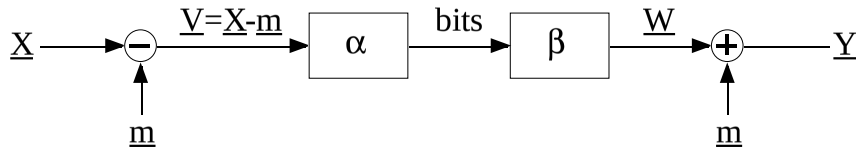
Notes:

- Property 1 follows from the law of total expectation.
- Properties 2-6 could also be derived directly from the orthogonality property of optimal estimators.
- The proof of Fact B is left as an exercise.

BASIC QUANTIZER TRANSFORMATIONS

1. Shifting

Given a quantizer characterized by k, M, L, S, C, Q, C_b, α, β, and an arbitrary k-dimensional vector $\underline{m}$, the *shifted* version of this quantizer is shown below:



This shifted quantizer is characterized by k, M, L, S', C', Q', C_b, α', β':

$$S' = \{S'_1,\ldots,S'_M\} = S' + \underline{m}\,, \qquad S'_i = S_i + \underline{m} = \{\underline{x}+\underline{m} : \underline{x} \in S_i\}$$

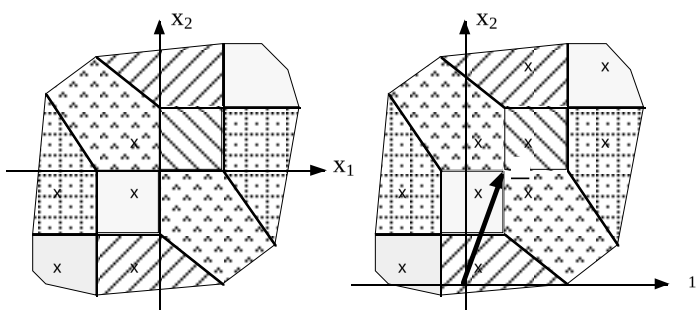
$$C' = \{\underline{w}'_1,\ldots,\underline{w}'_M\} = C' + \underline{m}\,, \qquad \underline{w}'_i = \underline{w}_i + \underline{m}$$

$$Q'(\underline{x}) = Q(\underline{x}-\underline{m}) + \underline{m}$$

$$\alpha'(\underline{x}) = \alpha(\underline{x}-\underline{m})$$

$$\beta'(i) = \beta(i) + \underline{m}$$

When deriving S' and C', it easiest to look at the decoder.
Basically the new quantizer has been "shifted" by $\underline{m}$, as illustrated below.



A quantizer. The quantizer shifted by $\underline{m}$.

Facts:

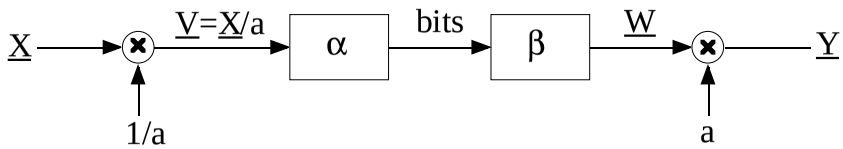
1. The rate of the shifted quantizer is the same as the rate of original quantizer.
2. The MSE of the shifted quantizer applied to $\underline{X}$ is the same as the MSE of the original quantizer applied to $\underline{V}$.
3. If k, M, L, S, C is optimal for random vector $\underline{V}$ with pdf $p_{\underline{V}}(\underline{v})$, then the shifted quantizer k, M, L, S', C' is optimal for the random vector $\underline{X} = \underline{V}+\underline{m}$, with pdf $p_{\underline{X}}(\underline{x}) = p_{\underline{V}}(\underline{x}-\underline{m})$.

   One can prove this by showing that if there were a better quantizer for $\underline{X}$, then one could find a better quantizer for $\underline{V}$, which would contradict the assumed optimality of the original quantizer for $\underline{V}$.

These facts imply that one can optimize a quantizer for a zero mean random vector, and then shift it to make it optimal for a random vector with mean $\underline{m}$.

2. Scaling

Given a quantizer characterized by k, M, L, S, C, Q, C_b, α, β, and a positive scale factor a, the *scaled* version of this quantizer is shown below:



This scaled quantizer is characterized by k, M, L, S', C', Q', C_b, α', β':

$$S' = \{S'_1,\ldots,S'_M\} = aS', \qquad S'_i = aS_i = \{a\underline{x} : \underline{x} \in S_i\}$$

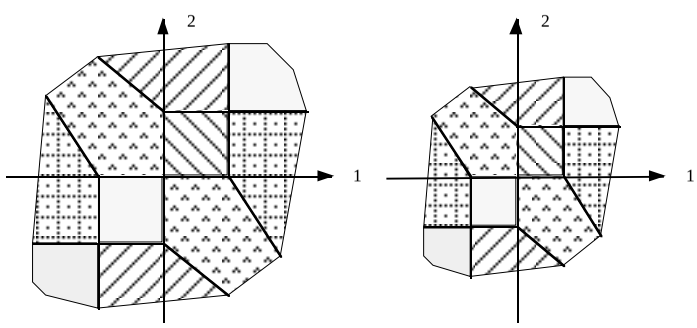
$$C' = \{\underline{w}'_1,\ldots,\underline{w}'_M\} = aC', \qquad \underline{w}'_i = a\underline{w}_i$$

$$Q'(\underline{x}) = a\,Q(\underline{x}/a)$$

$$\alpha'(\underline{x}) = \alpha(\underline{x}/a)$$

$$\beta'(i) = a\beta(i)$$

When deriving S' and C', it easiest to look at the decoder.
Basically the new quantizer has been "scaled" by a, as illustrated below.



A quantizer. The quantizer scaled by a<1.

Facts:

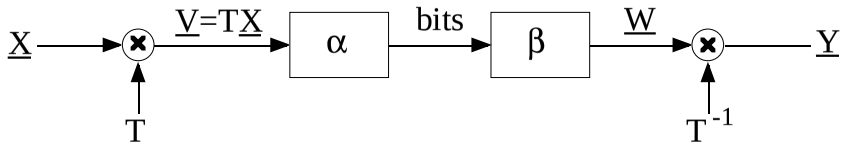
1. The rate of the scaled quantizer is the same as rate of the original quantizer.
2. The MSE of the scaled quantizer applied to $\underline{X}$ is a^2 times the MSE of the original quantizer applied to $\underline{V}$.
3. If k, M, L, S, C is optimal for random vector $\underline{V}$ with pdf $p_{\underline{V}}(\underline{v})$, then the scaled quantizer k, M, L, S', C' is optimal for the random vector $\underline{X} = a\underline{V}$, with pdf $p_{\underline{X}}(\underline{x}) = \frac{1}{a} p_{\underline{V}}(\underline{x}/a)$.

   One can prove this by showing that if there were a better quantizer for $\underline{X}$, then one could find a better quantizer for $\underline{V}$, which would contradict the assumed optimality of the original quantizer for $\underline{V}$.

Therefore one can optimize a quantizer for a unit variance random vector, and then scale it to make it optimal for a random vector with some other variance.

3. Linear Transforming

This is a kind of vector generalization of scaling. Given a quantizer characterized by k, M, L, S, C, Q, C_b, α, β, and a k×k *orthogonal matrix*[6] T, the *transformed* version of this quantizer is shown below:



This transformed quantizer is characterized by k, M, L, S', C', Q', C_b, α', β':

$$S' = \{S'_1,\ldots,S'_M\} = T^{-1}S', \qquad S'_i = T^{-1}S_i = \{T^{-1}\underline{x} : \underline{x} \in S_i\}$$

$$C' = \{\underline{w}'_1,\ldots,\underline{w}'_M\} = T^{-1}C', \qquad \underline{w}'_i = T^{-1}\underline{w}_i$$

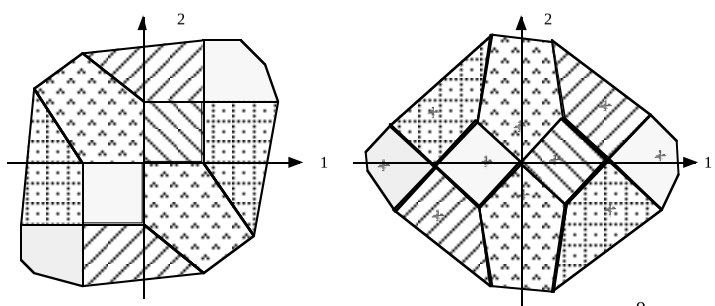
$$Q'(\underline{x}) = T^{-1}\,Q(T\underline{x})$$

$$\alpha'(\underline{x}) = \alpha(T\underline{x})$$

$$\beta'(i) = T^{-1}\beta(i)$$

When deriving S' and C', it easiest to look at the decoder.
Basically the new quantizer has been rotated by T^{-1}, as illustrated below.

[6] An orthogonal matrix is a matrix having the property that $\|T\underline{x}\| = \|\underline{x}\|$ for all $\underline{x}$. Equivalently, its rows are orthonormal. Equivalently, its column are orthonormal. Here we view $\underline{x}$ as a column vector. The determinant of an orthogonal matrix is one.



A quantizer. The quantizer rotated by 45°.

Facts:

1. The rate of transformed quantizer is same as rate of the original quantizer.
2. The MSE of the transformed quantizer applied to $\underline{X}$ is the same as the MSE of the original quantizer applied to $\underline{V}$.
3. If k, M, L, S, C is optimal for random vector $\underline{V}$ with pdf $p_{\underline{V}}(\underline{v})$, then the scaled quantizer k, M, L, S', C' is optimal for the random vector $\underline{X} = T^{-1}\underline{V}$, with pdf $p_{\underline{X}}(\underline{x}) = p_{\underline{V}}(T\underline{x})$.

   One can prove this by showing that if there were a better quantizer for $\underline{X}$, then one could find a better quantizer for $\underline{V}$, which would contradict the assumed optimality of the original quantizer for $\underline{V}$.

Therefore one can optimize a quantizer for one random vector, and then rotate it to make it optimal for a random vector with a rotated probability distribution.

Note:

In conventional "transform coding", such as JPEG, a great deal of attention is paid to choosing the orthogonal matrix. Why is this? Fact 3 seems to suggest that the choice of orthogonal matrix doesn't matter.

4. Nonlinear transformations

Before the days of cheap digital hardware, nonlinear transforms were used to to make nonuniform scalar quantizers from a uniform scalar quantizer.

BASIC QUANTIZER COMBINATIONS

1. The Product of Two Quantizers

Suppose we are given two vector quantizers

$k, M_k, L_k, S_k, C_k, Q_k, C_{b,k}, \alpha_k, \beta_k$ and $m, M_m, L_m, S_m, C_m, Q_m, C_{b,m}, \alpha_m, \beta_m$

The *product* of these is a quantizer with dimension $k+m$ that uses the first quantizer on the first k samples of $\underline{x}$ and the second quantizer on the last m samples. (It can be veiwed as *time-sharing* the two.) It is characterized by

$$k+m, \quad M = M_k M_m, \quad L = L_k + L_m, \quad S = S_k \times S_m, \quad C = C_k \times C_m,$$

$$Q(\underline{x}) = (Q(x_1,\ldots,x_k), Q(x_{k+1},\ldots,x_{k+m})), \quad C_b = C_{b,k} \times C_{b,m},$$

$$\alpha(\underline{x}) = (\alpha_k(x_1,\ldots,x_k), \alpha_m(x_{k+1},\ldots,x_{k+m})), \quad \beta(\underline{z}) = (\beta_k(z_1,\ldots,z_{L_k}), \beta_m(z_{L_k+1},\ldots,z_{L_k+L_m}))$$

Note: "$\times$" denotes Cartesian product. The Cartesian product $S_k \times S_m$ of partitions S_k and S_m is defined to be the partition consisting of the Cartesian products of each cell of S_k with each cell of S_m.

Facts:

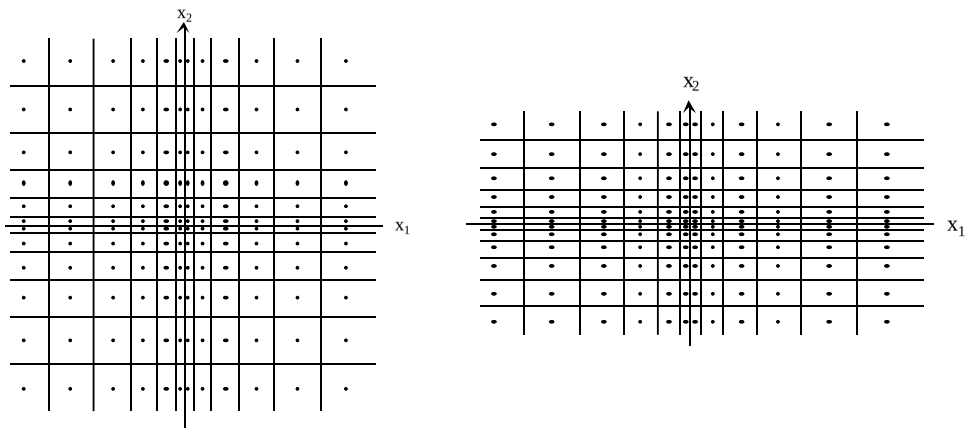
1. The rate of the product quantizer is

$$R = \frac{L_k+L_m}{k+m} = \frac{k}{k+m} R_k + \frac{m}{k+m} R_m$$

2. The MSE of the product quantizer is

$$D = \frac{k}{k+m} D_k + \frac{m}{k+m} D_m$$

Examples of the product of two scalar quantizers



Notes:

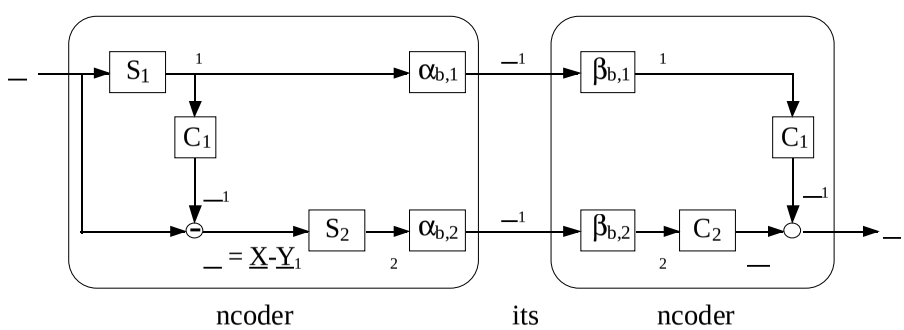
1. In the same way, one can form the product of three or more quantizers.
2. Product quantizers are not really a new VQ method. Rather it is principally a way to view lower dimensional quantizers in higher dimensions. This will make it possible to make reasonable comparisons of quantizers with different dimensions. For example, to compare a scalar quantizer to a 2-dimensional quantizer, we form the product of the scalar quantizer with itself. This does not change the rate or distortion. Then we can compare the characteristics of the product quantizer to those of the 2-dimensional quantizer.

2. Two-Stage Quantization

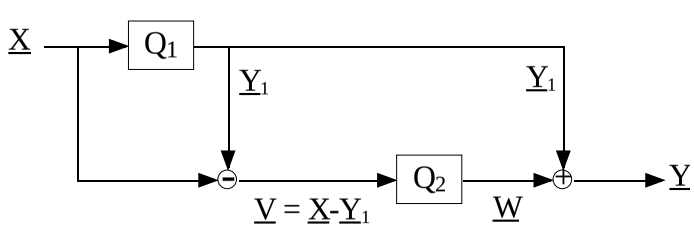
Suppose we are given two vector quantizers with the same dimension k

$$k, M_1, L_1, S_1, C_1, Q_1, C_{b,1}, \alpha_1, \beta_1 \text{ and } k, M_2, L_2, S_2, C_2, Q_2, C_{b,2}, \alpha_2, \beta_2.$$

Two-stage quantization is formed by interconnecting them as shown below:



The picture is simpler if we just show the quantization rule, as below:



The characteristics of the two-stage quantizer are:

$$k,\ M = M_1 M_2,\ L = L_1+L_2,\ C = C_1 \oplus C_2 = \{\underline{w}_1+\underline{w}_2 : \underline{w}_1 \in C_1, \underline{w}_2 \in C_2\} = \text{direct sum}$$

$$Q(\underline{x}) = Q_1(\underline{x}) + Q_2(\underline{x}-Q_1(\underline{x})),\ C_{b,1} = C_{b,1} \times C_{b,2},$$

$$\alpha(\underline{x}) = (\alpha_1(\underline{x}), \alpha_2(\underline{x}-Q_1(\underline{x})),\quad \beta(\underline{z}_1,\underline{z}_2) = \beta(\underline{z}_1) + \beta(\underline{z}_2)$$

The partition S is not as simple to describe. If $S_{i,j}$ denotes the cell corresponding to the codevector $\underline{w}_{1,i} + \underline{w}_{2,j}$, then $S_{i,j} = S_{1,i} \cap (S_{2,j}+\underline{w}_{1,i})$.

Facts:

1. The rate of the two-stage quantizer is

$$R = R_1 + R_2$$

2. The MSE of the two-stage quantizer is the MSE of the second stage, i.e.

$$D = \frac{1}{k} E\, ||\underline{V}-\underline{W}||^2$$

One can show this by showing that $\underline{X}-\underline{Y} = \underline{V}-\underline{W}$.

Notes:

1. Two-stage quantization does a coarse quantization followed by a fine quantization.
2. Two-stage can be generalized to three-stage, four-stage, etc.
3. Many lossy source coding systems use the two-stage or multistage approach.

3. Union Codebook Quantizer

Suppose we are given two quantizers with the same dimension k

k, M_1, L_1, S_1, C_1, Q_1, $C_{b,1}$, α_1, β_1 and k, M_2, L_2, S_2, C_2, Q_2, $C_{b,2}$, α_2, β_2.

The *union codebook quantizer* has codebook

$$C = C_1 \cup C_2$$

There are two encoding rules (and corresponding partitions) to consider:

a. Optimal encoding with a Voronoi partition for C.

b. Try each encoder and use the best one. Send one bit to indicate which encoder is used. That is, compute $||\underline{x}-Q_1(\underline{x})||$ and $||\underline{x}-Q_2(\underline{x})||$, and then

$$\alpha(\underline{x}) = \begin{cases} (0,\alpha_1(\underline{x})), & \text{if } ||\underline{x}-Q_1(\underline{x})|| \le ||\underline{x}-Q_2(\underline{x})|| \\ (1,\alpha_2(\underline{x})), & \text{if } ||\underline{x}-Q_2(\underline{x})|| < ||\underline{x}-Q_1(\underline{x})|| \end{cases}$$

Facts:

1. Upper bound to the rate:

$$R \le \max\{R_1,R_2\} + \frac{1}{k}$$

2. Upper bound to distortion:

$$D \le \min\{D_1,D_2\}$$