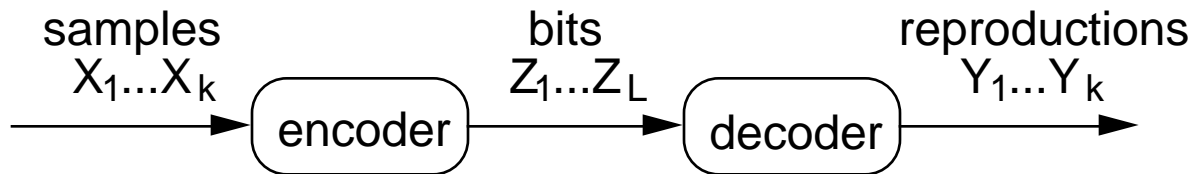


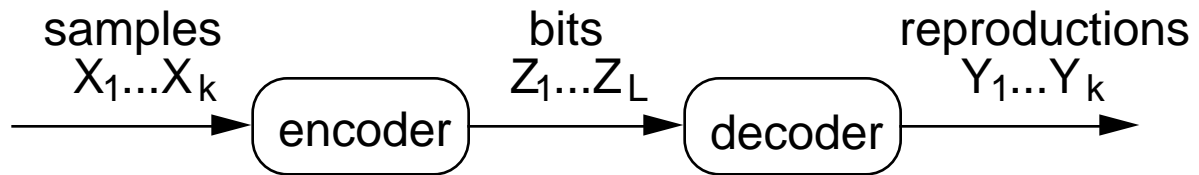
Vector Quantization (VQ)

Introduction

Consider an arbitrary fixed-rate lossy source code that operates independently on blocks (vectors) of k real-valued samples: k samples into encoder, L bits out



- source coder = encoder + decoder
- encoder and decoder are described by functions called encoding rule and decoding rule, respectively
- the sets of all possible encoder outputs and all possible decoder outputs play important roles
- the partition induced on the space of k -dimensional input vectors plays an important role
- a lossy source code that operates independently on fixed-length blocks, producing fixed-length blocks of bits is called a *fixed-rate (memoryless) vector quantizer (VQ)*
- fixed-rate VQ is a very general paradigm that includes many lossy source codes as special cases, e.g. fixed-rate transform coding. Since it is quite generable and also *analyzable*, it provides an excellent framework for studying lossy source codes.
- JPEG has variable, not fixed, rate. Except for the encoding of dc coefficients, it operates independently on blocks of 64 pixels. (It's a variable-rate VQ with memory.)



Key Characteristics (high-level, input-output)

Dimension: k

Encoding rule: $e: R^k \rightarrow \{0,1\}^L$

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

$$Z_{2L+1} \dots Z_{3L} = e(X_{2k+1} \dots X_{3k}), \dots$$

Binary codebook:

$$C_b = \{e(\underline{x}): \underline{x} \in R^k\} = \{\underline{c}_1, \underline{c}_2, \dots, \underline{c}_M\}$$

where $\underline{c}_i = (c_{i1}, c_{i2}, \dots, c_{iL}) = i$ th binary codeword

Size of code: M

Decoding rule $d: C_b \rightarrow R^k$

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

$$Y_{2k+1} \dots Y_{3k} = e(Z_{2L+1} \dots Z_{3L}), \dots$$

(Quantization) Codebook:

$$C = \{d(\underline{c}_1), d(\underline{c}_2), \dots, d(\underline{c}_M)\} = \{\underline{w}_1, \underline{w}_2, \dots, \underline{w}_M\}$$

where $\underline{w}_i = (w_{i1}, \dots, w_{ik}) = i$ th codevector

(code/reproduction, vector/point)

Quantization rule: $Q: R^k \rightarrow R^k$

$$Q(\underline{x}) = d(e(\underline{x})) = \text{reproduction produced by decoder for } \underline{x}$$

(Quantizing) Partition: $S = \{S_1, S_2, \dots, S_M\}$

where $S_i = \{\underline{x} \in R^k : e(\underline{x}) = \underline{c}_i\} = \{\underline{x} \in R^k : Q(\underline{x}) = \underline{w}_i\}$
 = i th (quantizing) cell

(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*.)

Summary: A VQ is characterized by

Dimension: k

Size: M

Encoding rule: e

characterized by

Partition: $S = \{S_1, S_2, \dots, S_M\}$ and

Binary codebook: $C_b = \{\underline{c}_1, \underline{c}_2, \dots, \underline{c}_M\}$

Decoding rule: d

characterized by

Codebook $C = \{\underline{w}_1, \underline{w}_2, \dots, \underline{w}_M\}$

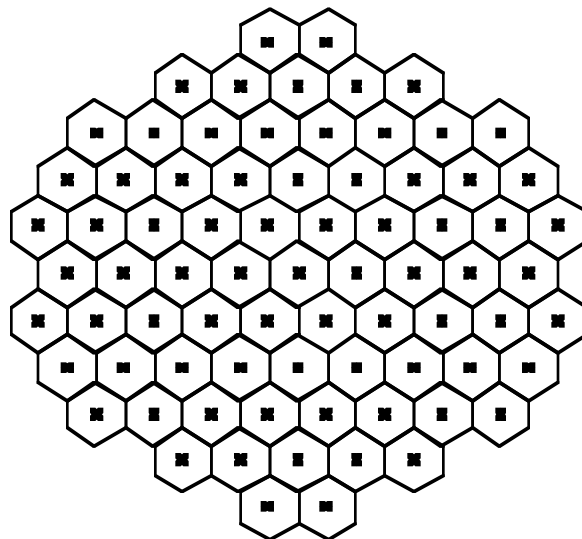
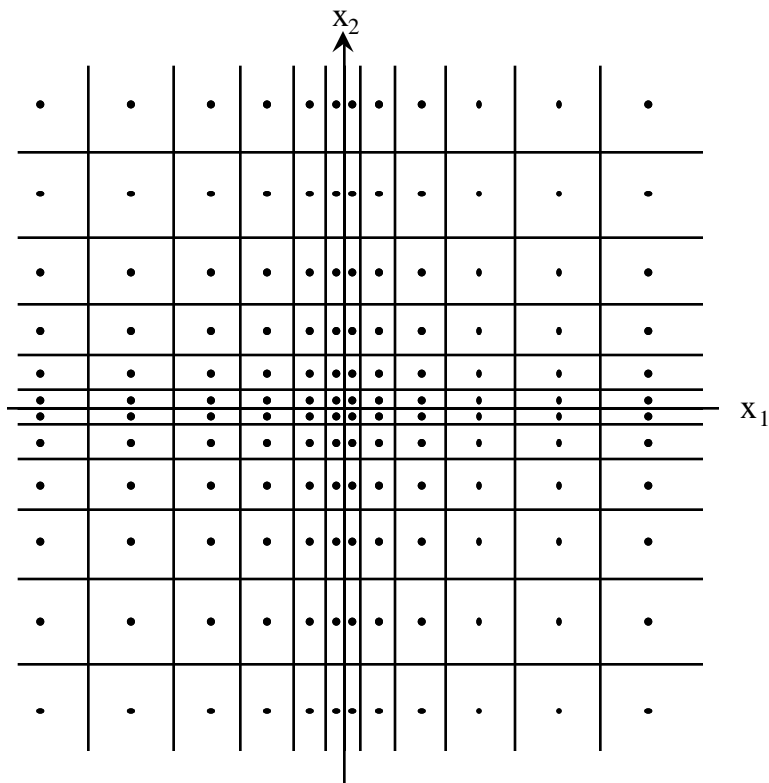
Quantization rule: Q

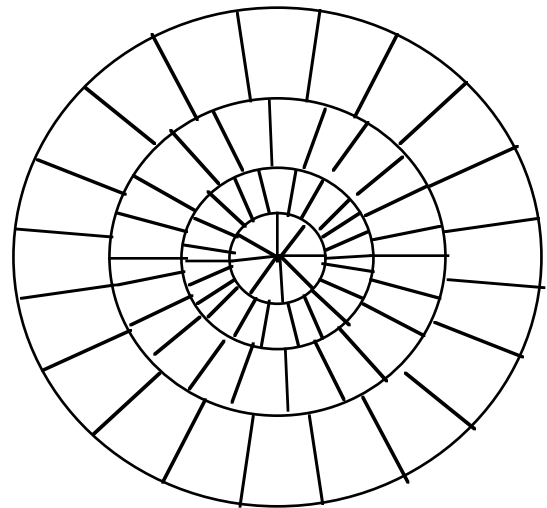
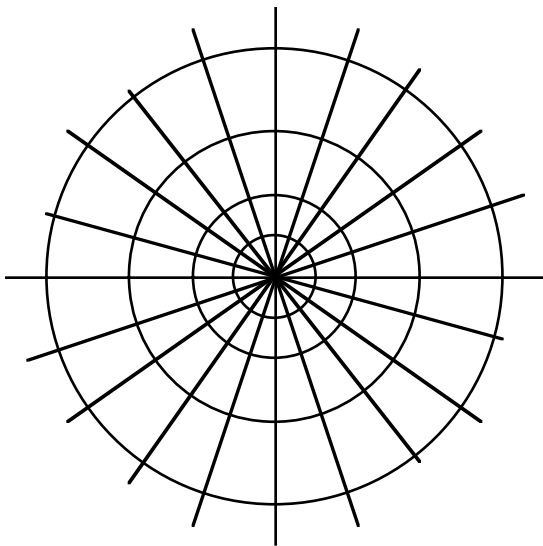
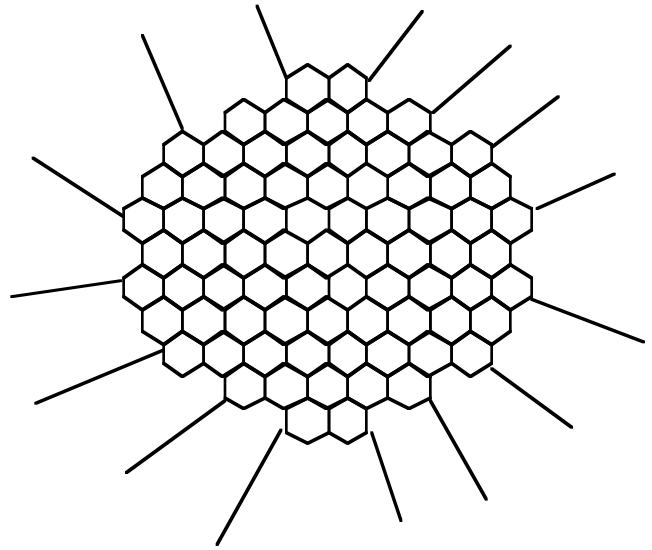
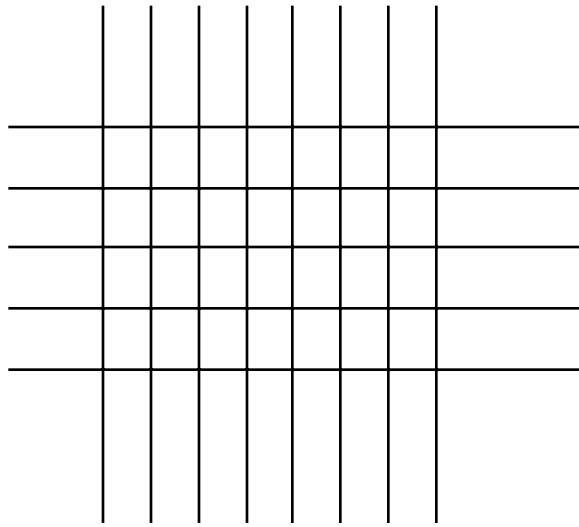
characterized by S and C

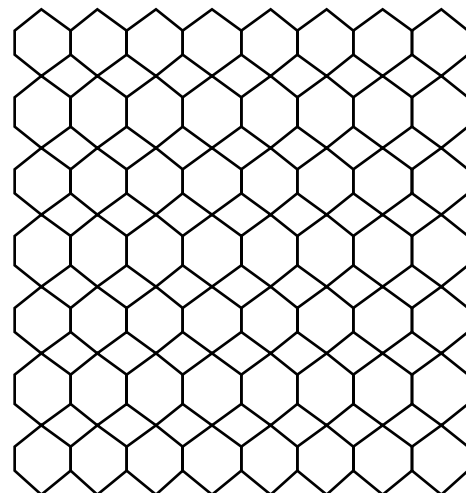
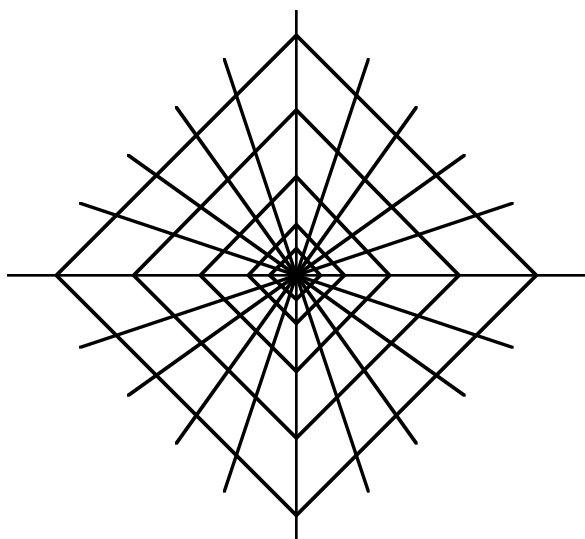
$Q(\underline{x}) = \underline{w}_i$ when $\underline{x} \in S_i$

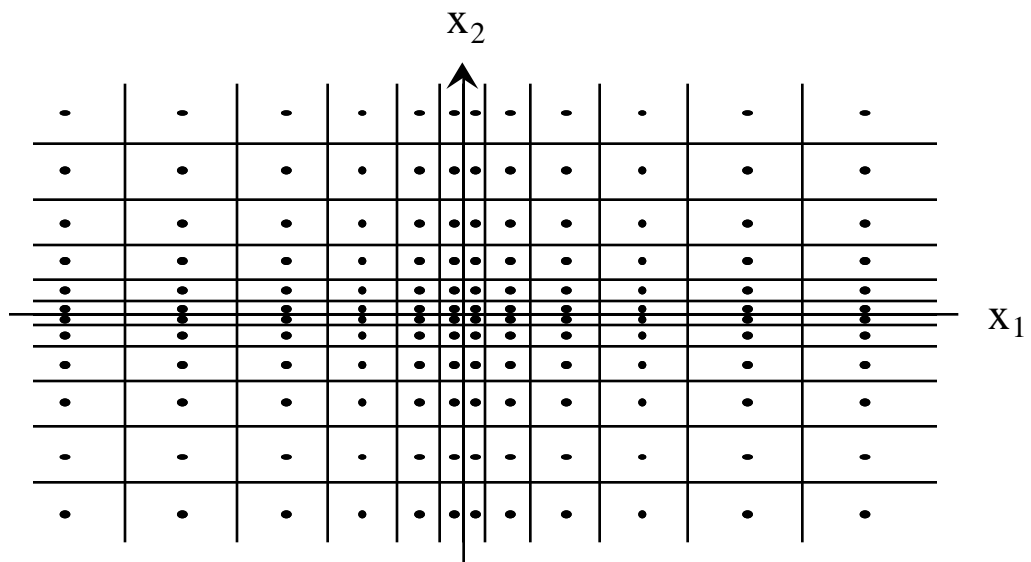
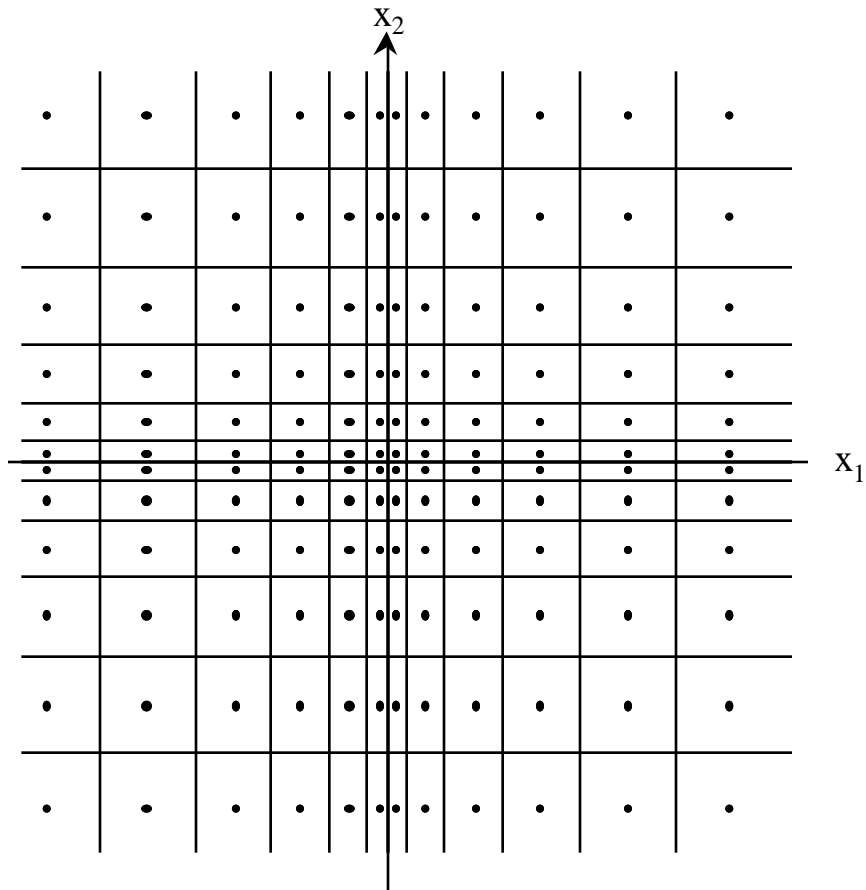
(You must learn to use this terminology and notation.)

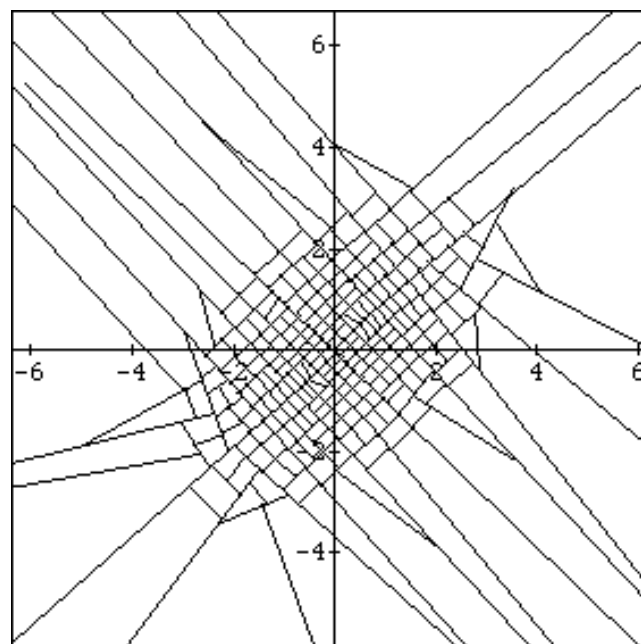
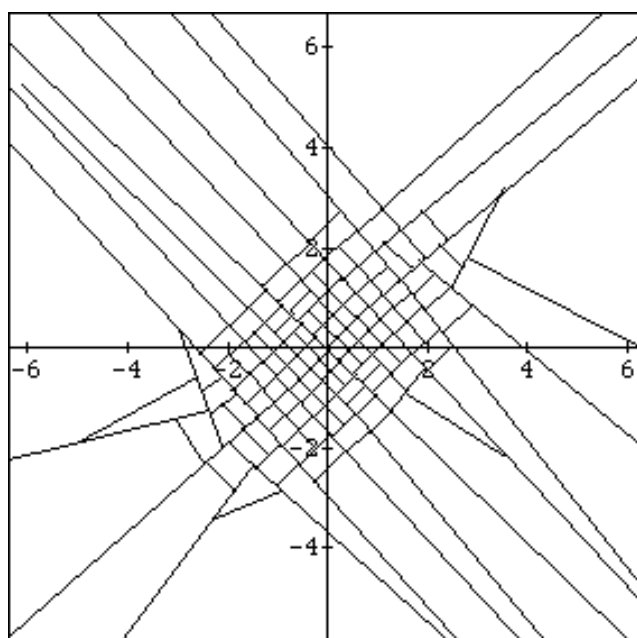
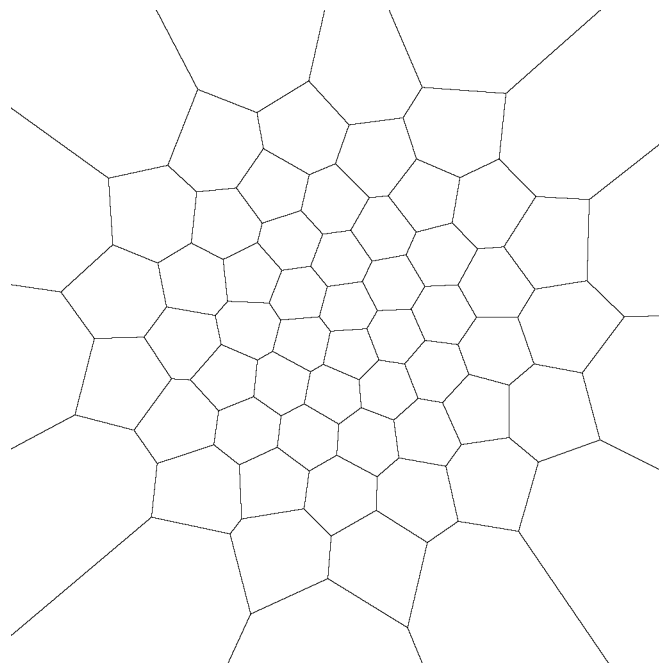
Examples (in $k=2$ dimensions)

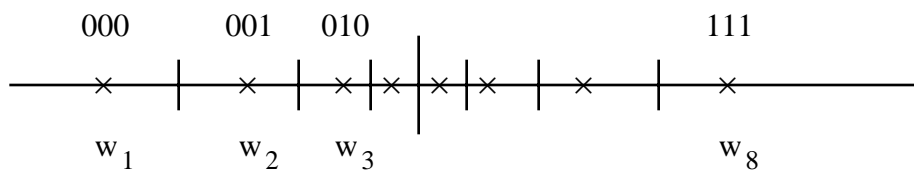










Scalar Quantizer ($k=1$)

Performance

Rate

$$R = \frac{L}{k}$$

$$= \begin{cases} \frac{1}{k} \log_2 M, & \text{if } M = \text{power of } 2 \\ \frac{1}{k} \lceil \log_2 M \rceil, & \text{if } M \neq \text{power of } 2 \text{ \& one index coded at a time} \\ \frac{1}{mk} \lceil \log_2 M^m \rceil \cong \frac{1}{k} \log_2 M, & \text{if } M \neq \text{power of } 2 \\ & \text{\& } m \text{ indices coded at a time} \end{cases}$$

Accordingly, we assume

$$R = \frac{1}{k} \log_2 M \quad \text{unless there is need to be picky.}$$

units: bits/sample

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

Distortion

$$\begin{aligned}
 D &= \text{MSE} = \text{mean squared error (normalized by dimension)} \\
 &= \frac{1}{k} \sum_{i=1}^k E(X_i - Y_i)^2 = \frac{1}{k} E \sum_{i=1}^k (X_i - Y_i)^2 \\
 &= \frac{1}{k} E \|\underline{X} - \underline{Y}\|^2 = \frac{1}{k} E \|\underline{X} - Q(\underline{X})\|^2 = \frac{1}{k} \int \|\underline{x} - Q(\underline{x})\|^2 f_{\underline{X}}(\underline{x}) d\underline{x} \\
 &= \frac{1}{k} \sum_{i=1}^k \int_{S_i} \|\underline{x} - \underline{w}_i\|^2 f_{\underline{X}}(\underline{x}) d\underline{x}
 \end{aligned}$$

where

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

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

Source vector

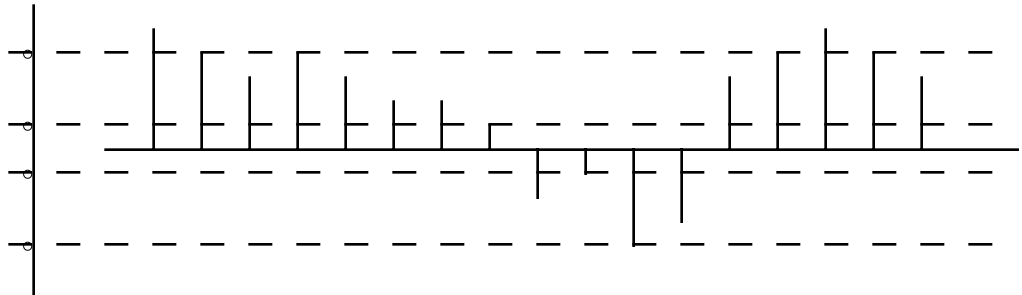
We assume \underline{X} is modeled as a vector of continuous random variables $\underline{X} = (X_1, \dots, X_k)$, whose probability distribution is characterized by a joint density denoted $f_{\underline{X}}(\underline{x})$ (or just $f(\underline{x})$).

When $\forall a$ Q operates on sequence of vectors, $\underline{X}_1, \underline{X}_2, \dots$, we usually assume these vectors come from a stationary random process.

Note: Distortion is determined by Q (alternatively, the partition and codebook). The specific encoding rule, decoding rule and binary codebook do not matter, except as they determine Q .

How VQ differs from SQ

Consider output of correlated source



Consider scalar quantization with four levels: $C_1 = \{-4, -1, 1, 4\}$.

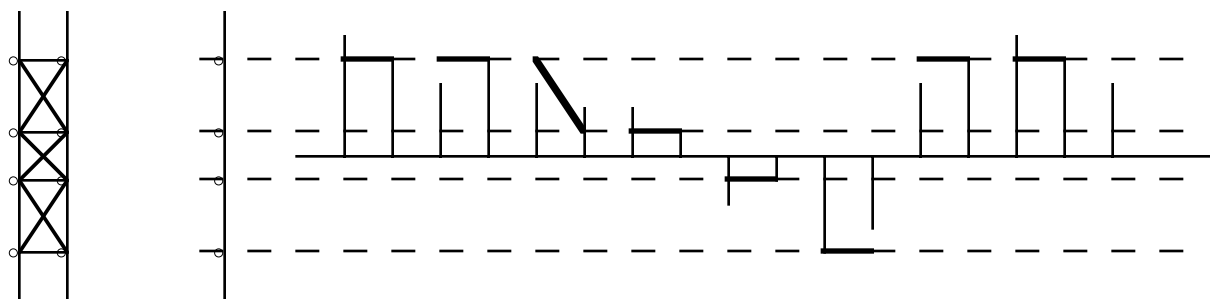
Since successive source samples are mostly quantized into adjacent levels, consider the following 2-dimensional VQ, with only 10 pairs of levels (codevectors)

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

Rather than the 16 pairs produced by scalar quantization,

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

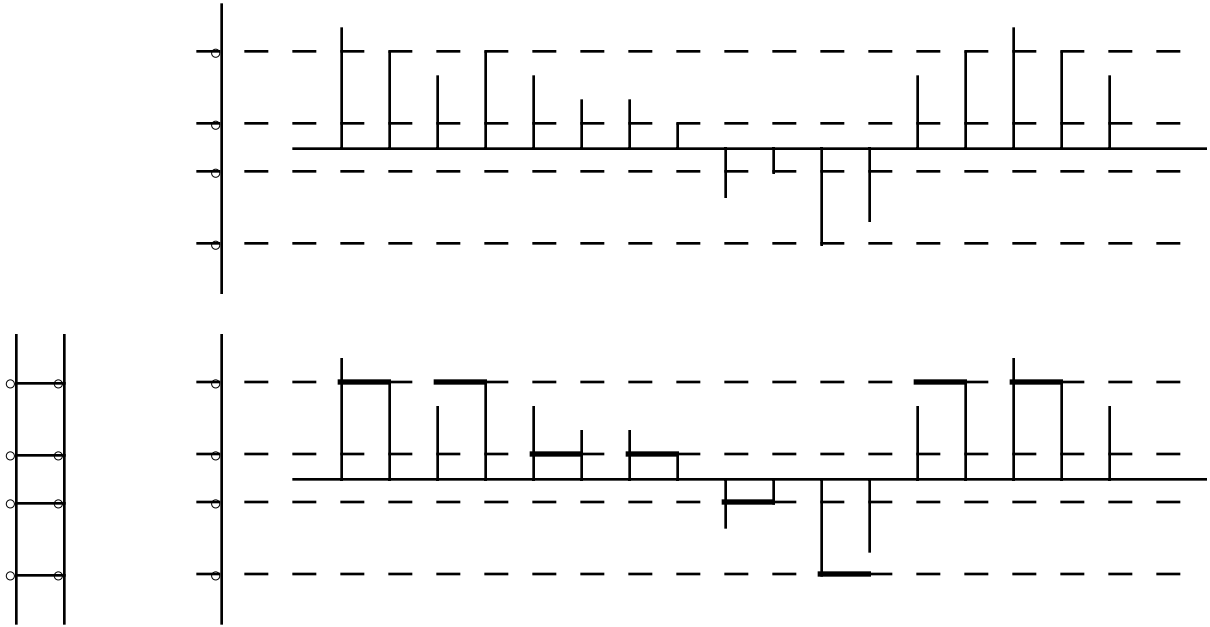
VQ output



Or consider the following 2-dimensional VQ, with codebook consisting only of 4 pairs of levels (codevectors),

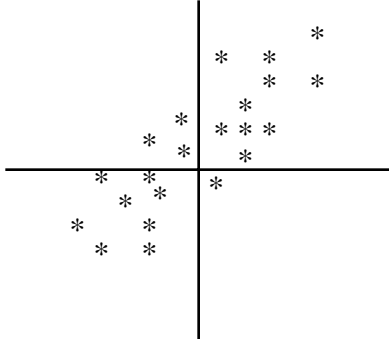
$$C_2 = \{(-4,-4),(-1,-1),(1,1),(4,4)\}$$

rather than the 16 pairs produced by scalar quantization,



Another view

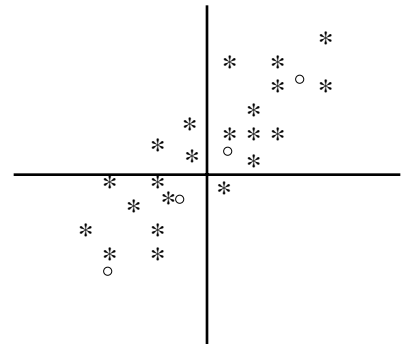
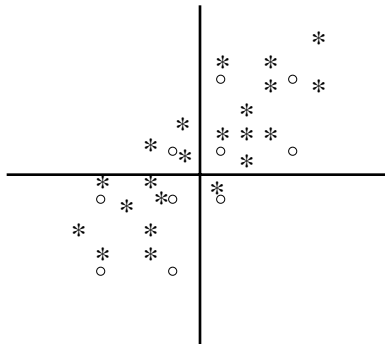
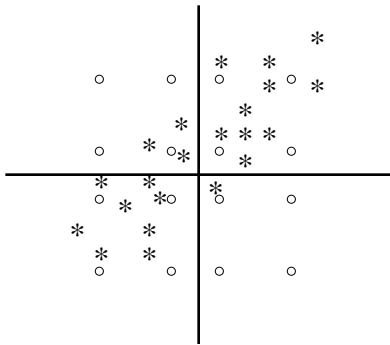
Scatter plot of typical (x_1, x_2) pairs



scalar quant. used twice

VQ with $M = 10$

VQ with $M = 4$



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 optimal 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 coverage:

Optimality properties of fixed-rate VQ's.

"Full search" encoding.

Generalized Lloyd iterative VQ design algorithms

Properties of optimal quantizers, e.g. $E \|Y\|^2 = E \|X\|^2 - E \|X-Y\|^2$

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

Optimality Properties (Useful in design and implementation)

Property 1: Given a codebook $C = \{\underline{w}_1, \dots, \underline{w}_M\}$, let

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

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

$$S_i \doteq V_i \text{ for each } i, \quad (*)$$

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

Interpretation/Derivation:

The role of the encoder is to control the decoder to produce the best reproduction among all that it can produce.

Therefore, if the source vector \underline{x} is closer to \underline{w}_i than to any other codevector, then to minimize MSE, \underline{x} should be quantized to \underline{w}_i ; i.e. \underline{x} should be in S_i .

This is Prop 1, except that Prop 1 recognizes that MSE is not affected if the S_i differs from V_i 's by a set of probability zero.

Clearly, a partition satisfying (*) has smallest possible MSE for the given set of codevectors; i.e. it cannot be substantively improved. Conversely, if a partition does not satisfy (*), it could be substantively improved, so it could not have smallest MSE.

Note: There will always be points not contained in any V_i , namely points for which there are two or more closest codevectors. However, the set of such points has zero volume because it is a $(k-1)$ -dimensional subset of R^k . Consequently, it has zero probability. Such points could be assigned to any cell, however, they would normally be assigned to the cell corresponding to one of the closest codevectors.

Notes:

- A partition S such that $S_i \supseteq V_i$ for all i is called a "Voronoi partition", and 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 of the subset

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

All T_i 's have probability zero.

- Ordinarily, we won't fuss about the sets of probability zero and the assignment of points that are equidistant between codevectors and simply say that "the optimal partition or the Voronoi partition is"

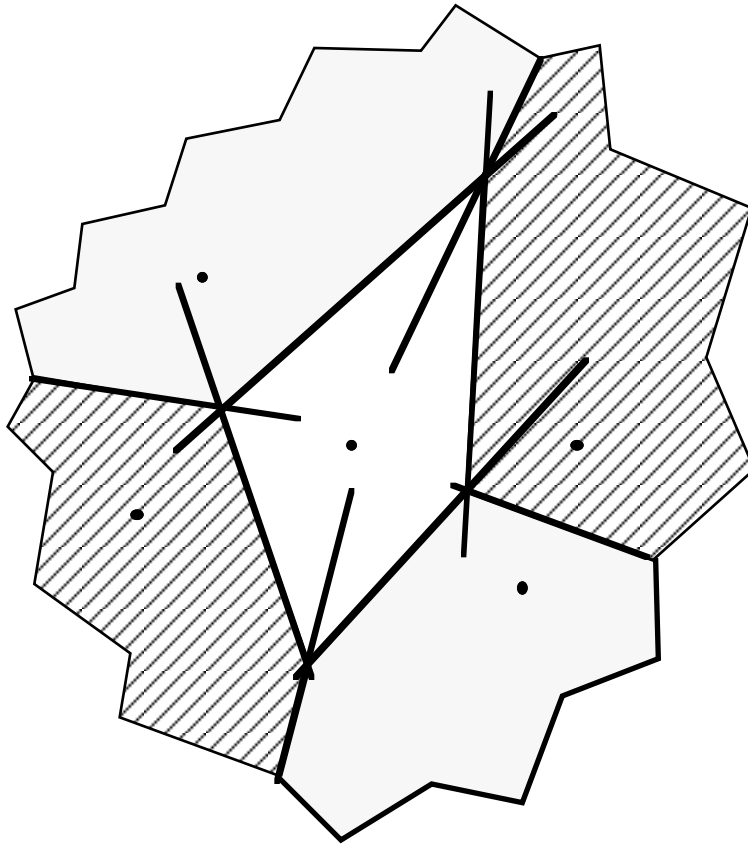
$$S_i = \{\underline{x} : \|\underline{x}-\underline{w}_i\| < \|\underline{x}-\underline{w}_j\|, \text{ for all } j \neq i\}$$

or

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

- The Voronoi partition can be found by drawing perpendicular bisectors between each pair of codevectors. These are $(k-1)$ -dimensional hyperplanes, each of which divides 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.

- 2-Dimensional Example:



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

Property 2: Given a partition $S = \{S_1, \dots, S_M\}$ and source density $f_{\underline{X}}(\underline{x})$, the codevectors that minimize MSE are the "centroids"

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

where

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

Interpretation/Derivation:

The role of the decoder is to make the best estimate of the source output \underline{X} , given its input, which is, effectively, knowledge of the cell in which \underline{X} lies.

When the decoder is told that \underline{X} lies in cell S_i , its output, namely \underline{w}_i , should be the minimum MSE estimate of \underline{X} given this knowledge, i.e. $\underline{w}_i = E[\underline{X}|\underline{X} \in S_i]$. Any other choice leads to larger MSE. (Recall that $E\|\underline{X}-\underline{c}\|^2$ is minimized by $\underline{c} = E\underline{X}$.)

If this is not already clear, it can be seen from the following:

$$\begin{aligned} D &= \sum_{i=1}^M \Pr(\underline{X} \in S_i) \int_{S_i} (\underline{x} - \underline{w}_i)^2 \frac{f_{\underline{X}}(\underline{x})}{\Pr(\underline{X} \in S_i)} d\underline{x} \\ &= \sum_{i=1}^M \Pr(\underline{X} \in S_i) \int (\underline{x} - \underline{w}_i)^2 f_{\underline{X}}(\underline{x}|\underline{X} \in S_i) d\underline{x} \\ &= \sum_{i=1}^M \Pr(\underline{X} \in S_i) E[(\underline{X} - \underline{w}_i)^2 | \underline{X} \in S_i] \end{aligned}$$

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

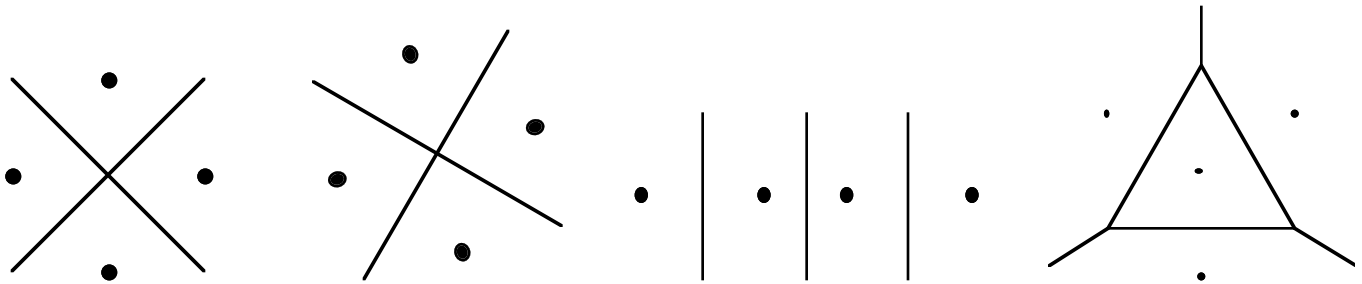
Note: The centroid of a convex cell is contained in the cell.

Corollary: Given a size M and a source density $f_{\underline{X}}(\underline{x})$, the best VQ (i.e. the one with smallest MSE) satisfies (*) and (**), which are called the "optimality properties" or "optimality criteria".

Notes:

- There may be more than one optimal quantizer.
- Even when there is only one optimal quantizer there may be more than one quantizer that satisfies that satisfies the optimality criteria, in which case the best quantizer is one of them.

Example: $k=2$, $M=4$, (X_1, X_2) is IID Gaussian. All of the following satisfy optimality criteria.



Which is best?

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 $\underline{w}_i + \varepsilon \underline{z}$ for an arbitrary vector \underline{z} and some scalar ε . If the VQ is locally optimal, then for any \underline{z} there is an ε_0 such that for all $\varepsilon \leq \varepsilon_0$, the perturbed VQ has the same or larger distortion as the original VQ. Any number of codevectors can be perturbed.

What about "sufficiently small perturbations" of the partition?

Imagine moving or stretching some boundary of some cell by an amount "proportional" to ε . Then there must exist some ε_0 such that for all $\varepsilon \leq \varepsilon_0$, 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_0 such that for all $\varepsilon \leq \varepsilon_0$, 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 a local optimal. The contrapositive of this is: Local 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, Unstructured Implementation of VQ

Full-Search Encoding

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

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

This method encodes uses the Voronoi partition S for C .
 C can be any codebook whatsoever.

Table-Lookup Decoding

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

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

This is the basic form of unstructured VQ. 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

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

encoding operations:

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

M-1 comparisons

The "curse of dimensionality"

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

The dimension-rate product kR is the key.

This is a big limitation. Generally,

$$kR \leq 10 \text{ or } 12.$$

While modern computers can implement larger codebooks, it turns out that designing them can be extremely difficult.

There are some faster encoding algorithms, but not have been proven to have less than exponential complexity.

Next, we consider some fast "search methods" for encoding. But these still have exponential complexity. Later we consider VQ's that are structured so as to allow reduced complexity encoding and/or decoding. They might not use a Voronoi partition.

Fast search methods:

Here we describe several fast methods for finding the closest codevector in a k -dimensional codebook $C = \{\underline{w}_1, \dots, \underline{w}_M\}$ to a given vector \underline{x} . All but the first of these obtain reduced computational complexity at the expense of increased storage.

References: A number of references describing fast codebook searches are listed in R. Gray and D. Neuhoff, "Quantization," *IEEE Trans. Inform. Thy*, Oct. 1998.

1. Partial distortion:

Suppose we have found that the closest codeword to \underline{x} among $\underline{w}_1, \dots, \underline{w}_{n-1}$ is at distance d from \underline{x} , and suppose we are now computing

$$\|\underline{x} - \underline{w}_n\|^2 = (x_1 - w_{n,1})^2 + (x_2 - w_{n,2})^2 + (x_3 - w_{n,3})^2 + \dots$$

by successively computing and accumulating the square terms.

If after a certain number of terms have been accumulated this sum should become larger than d^2 , then we know \underline{w}_n is not the closest codeword, so we need not compute any more of its terms, but move on to computing $\|\underline{x} - \underline{w}_n\|^2$.

Since most codewords are quite far from \underline{x} , this method frequently saves a lot of computation, typically, 25 to 75%.

2. Successive narrowing of search:

The distances between all pairs of codevectors are precomputed and stored in a table.

- a. Choose an initial codevector \underline{y} in some way.
- b. Eliminate from consideration all codevectors \underline{w}_i such that

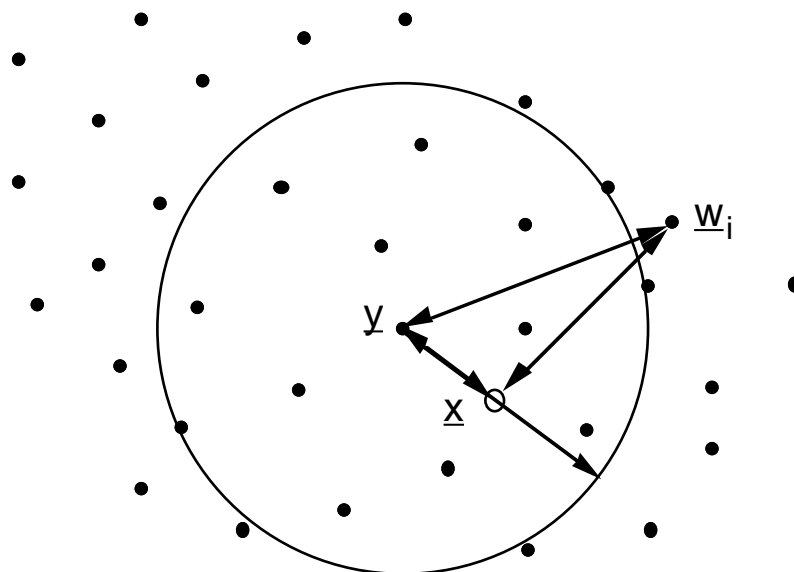
$$\|\underline{w}_i - \underline{y}\| > 2 \|\underline{x} - \underline{y}\|.$$

By the triangle inequality $\|\underline{w}_i - \underline{y}\| \leq \|\underline{w}_i - \underline{x}\| + \|\underline{x} - \underline{y}\|$, so that

$$\|\underline{w}_i - \underline{x}\| \geq \|\underline{w}_i - \underline{y}\| - \|\underline{x} - \underline{y}\| > \|\underline{x} - \underline{y}\|,$$

which shows that \underline{w}_i could not be the closest codeword.

- c. Successively search the codevectors not eliminated to find one closer to \underline{x} than \underline{y} , which then replaces \underline{y} .
- d. Eliminate from consideration all remaining codevectors such that $\|\underline{w}_i - \underline{y}\| > 2 \|\underline{x} - \underline{y}\|$.
- e. Repeat steps c. and d. until all codevectors have been considered or eliminated from consideration.



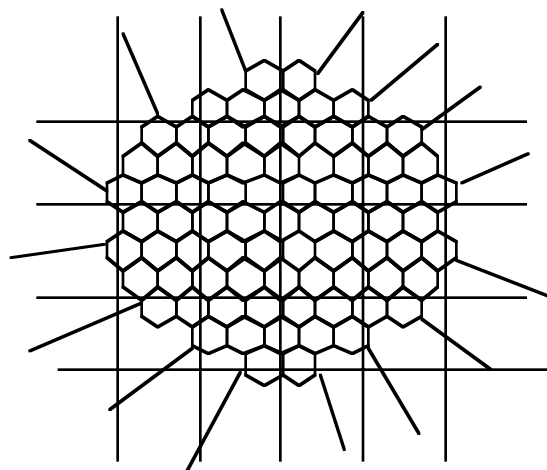
There are a number of variations of this method.

3. Coarse-to-fine search:

In this method, a very fast coarse quantization is done first. Then a full search is done on a smaller set.

Precompute a table: Select a scalar quantizer with rate R_s less than the rate of the desired VQ. Let $M_s = 2^{R_s}$. The table contains one row for each of the M_s^k possible sequences of k scalar levels. The row corresponding to a given sequence of levels contains the indices of all codevectors in C whose Voronoi regions intersect the k -dimensional rectangle formed by the Cartesian product of the scalar quantizer cells corresponding to the levels in the sequence.

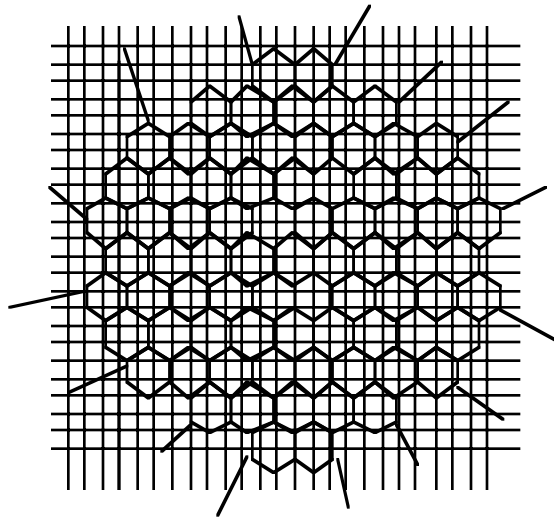
- Scalar quantize each component of \underline{x} with the scalar quantizer.
- Use the sequence of k scalar quantizer outputs to address the table and obtain a set of codevector indices.
- Compute the distance between \underline{x} and the codevectors whose indices were found in Step b.
- Output the closest codevector.



There are variations on this method where Step a. is replaced by some other coarse quantization method (e.g. a fast VQ).

4. Fine-to-coarse search.

This is just like coarse-to-fine, except the initial scalar quantization has sufficiently higher rate than the desired VQ that most of the k -dimensional rectangles described above are contained in one and only one Voronoi region of C . In this case, Step c. is eliminated in most cases, or reduced to just a few computations.



5. Hierarchical table lookup.

To be described later.

VQ Design Algorithms

Generalized Lloyd algorithms

These are iterative algorithms, in the spirit of Lloyd's scalar quantization algorithm. They seek a locally optimum quantizer by alternating between finding the best partition for the most recently found codebook and finding the best codebook for the most recently found partition. They stop when (*) and (**) are nearly satisfied; i.e. when a locally optimum VQ is approached. There are two basic types:

- A. Design from pdf
- B. Design from training sequence

A. Design from pdf.



Stop when the centroids and/or the distortion changes negligibly.

Convergence of the algorithm:

Since each step of finding Voronoi partition or find the the centroids does not increase distortion, the distortion of the algorithm is guaranteed to converge. Typically, the actual partition and codebook are converging to a local optimum (i.e. a VQ that satisfies (*) and (**)), which might or might not be the global optimum. However, it is conceivable that it might get into a cycle where, for example, it alternates between two different quantizers.

Initial codebook:

There are many possible choices for the initial codebook C_0 . Some commonly suggested possibilities include:

- (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 an optimal scalar quantizer with $M^{1/k}$ levels.
- (c) The set formed by adding an additional 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.

This algorithm is seldom used

Running this algorithm is often inhibited by the fact that the pdf of the source vector is not known. Moreover, even it is known, working with a k -dimensional pdf (e.g. computing the k -dimensional integrals in centroid calculations) is prohibitively complex. Therefore, in practice VQ's are almost always designed by training sequence methods such as that described next.

B. Design from training sequence. (LBG algorithm)

We have a version of this algorithm available.

Given training sequence $\underline{t}_1, \underline{t}_2, \dots, \underline{t}_N$, i.e. a representative sequence of k -dimensional vectors from the source.

Choose an initial codebook $C = \{\underline{w}_1, \dots, \underline{w}_M\}$.

Iterate the following two steps until the centroids change little and/or the distortion changes little:

1. Find Voronoi partition:

$$\tilde{S}_i = \{t_j : t_j \text{ closer to } \underline{w}_i \text{ than to any other codevector}\}$$

2. Find empirical centroids.

$$\tilde{\underline{w}}_i = \frac{1}{N_i} \sum_{j: t_j \in \tilde{S}_i} t_j, \quad \text{where } N_i = \# \text{ training vectors in } \tilde{S}_i.$$

Step 1 is implemented by constructing a table in which each training sequence vector is labeled with the index of the codevector to which it is closest.

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

Alternately, one may iterate just the following single step: given $\underline{w}_1, \dots, \underline{w}_M$, design new centroids via

$$\tilde{\underline{w}}_i = \frac{1}{N_i} \sum_{j: t_j \text{ closest to } \underline{w}_i} t_j, \quad \text{where } N_i = \# \text{ train'g vect's closest to } \underline{w}_i$$

In this case, one successively considers each training vector t_j , finds which codevector, say \underline{w}_i , it is closest to via a full search, then increments a counter that is computing N_i and adds t_j to an accumulator that is computing the sum that enters into $\tilde{\underline{w}}_i$.

Initial codebook:

The same choices as with Algorithm A are available.

Convergence of the algorithm:

Since each step of the algorithm maintains or decreases the training distortion, i.e. the distortion measured on the training sequence itself, the training distortion will necessarily converge. Typically, the partition and codebook generated by the algorithm are converging to a local optimum, i.e. to a pair that satisfy the empirical versions of (*) and (**). Since there are only a finite number of distinct partitions of the given training sequence, after a finite number of steps the algorithm must necessarily reach a local optimum after which it does not change, or it must cycle repeatedly through some finite number of partition-codebook pairs. However, the algorithm is usually stopped long before either of these occur. Because the algorithm deals only with a finite set of training vectors, there tend to be more local minimum than with algorithms based that work directly with the pdf such as Algorithm A. Therefore, it is usually wise to rerun the algorithm with several different choices of initial codebook.

Training distortion vs. actual distortion:

The actual distortion of the quantizer designed by this algorithm ($D = E(\underline{X} - Q(\underline{X}))^2$) is greater than the training distortion. To see the cause of this inequality, 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 might be far from the minimum. This inequality becomes smaller as N increases, but is nevertheless usually significant. Because training distortion can be substantially smaller than

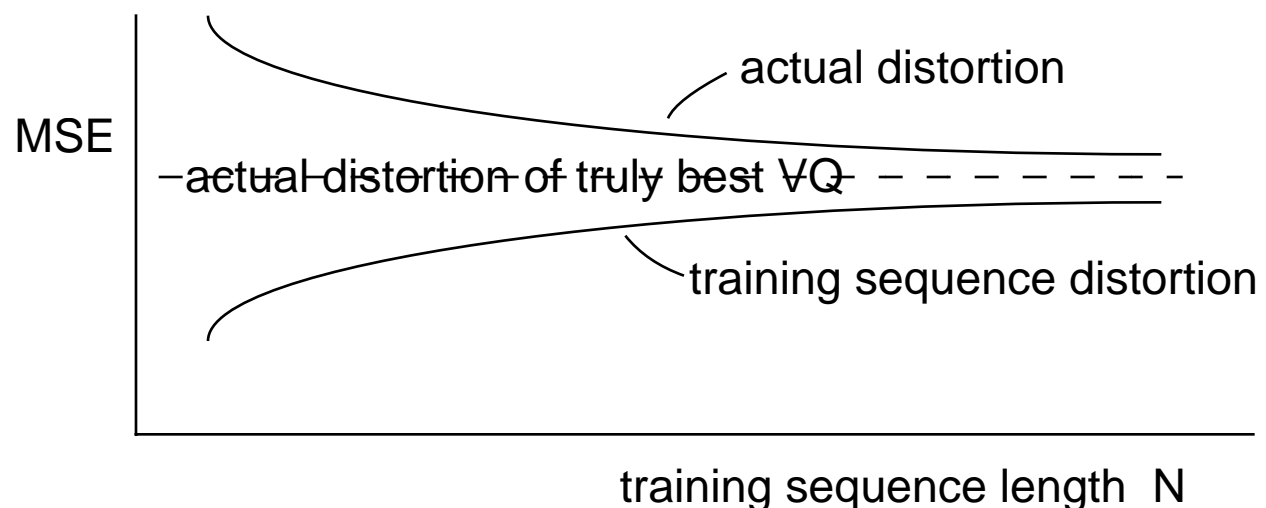
actual distortion, it is customary to estimate the actual distortion by running the VQ on a test sequence that is distinct from the training sequence.

Training sequence length:

An important question in VQ design is how large to make the training sequence length N . There are two issues: the VQ designed by the algorithm becomes better as N is made larger, and the training distortion becomes a more accurate estimate of actual distortion. The convergence of the actual and training distortion with increasing N are illustrated in the figure below. A typical rule of thumb is that to design a good VQ, N should be at least 50 M and larger is better.

Significantly larger N is needed if training distortion is to be used as an estimate of actual distortion.

Since it seems that for estimating distortion, a test sequence need not be as long as the training sequence required to make training distortion a good approximation to actual distortion, it makes sense to use such test sequences. A conservative strategy is to choose N large enough that the training and test sequence distortion are reasonably close.



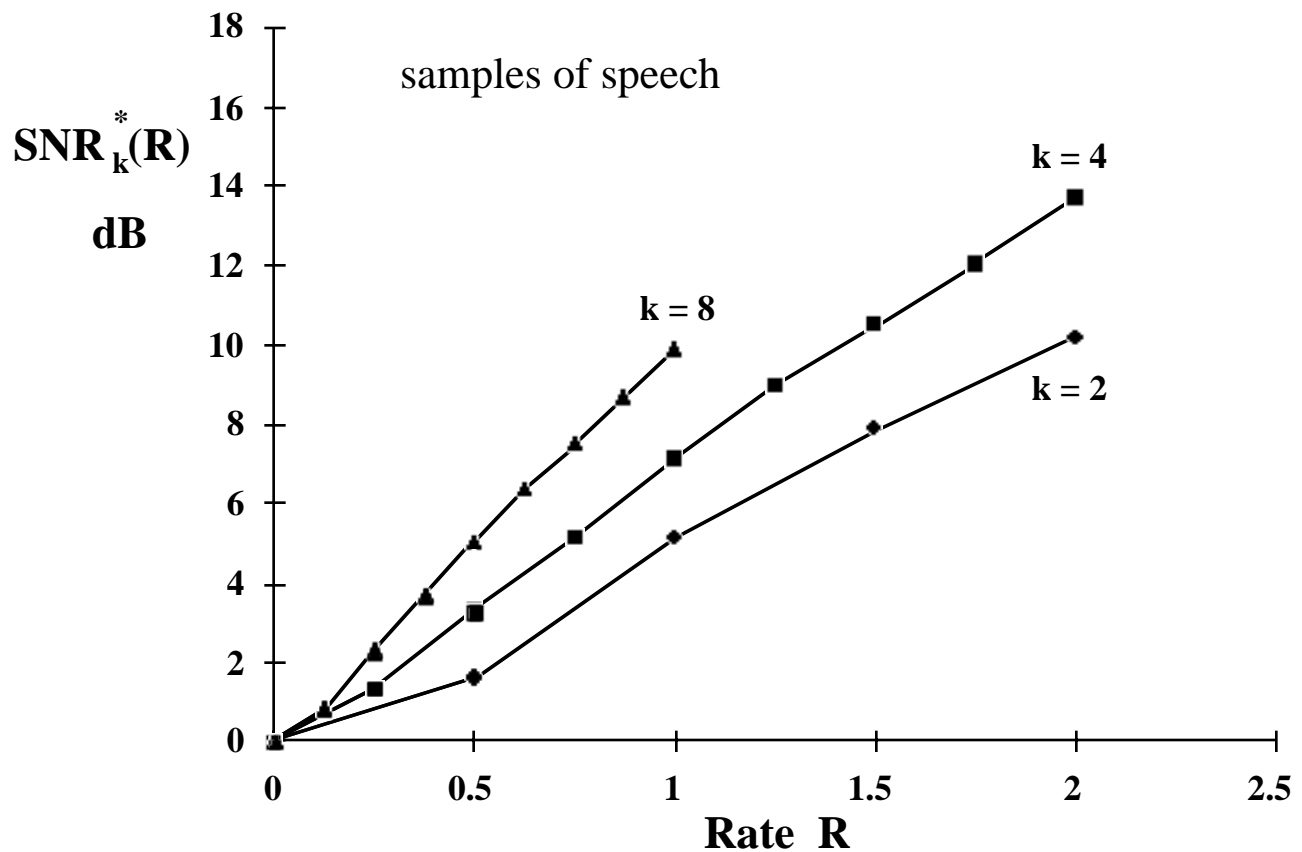
Complexity of VQ design:

The above design algorithm essentially 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. (I can't recall having seen a VQ designed with $kR > 14$). Thus there are dimension-rate products e.g. $kR = 15$ 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 described 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

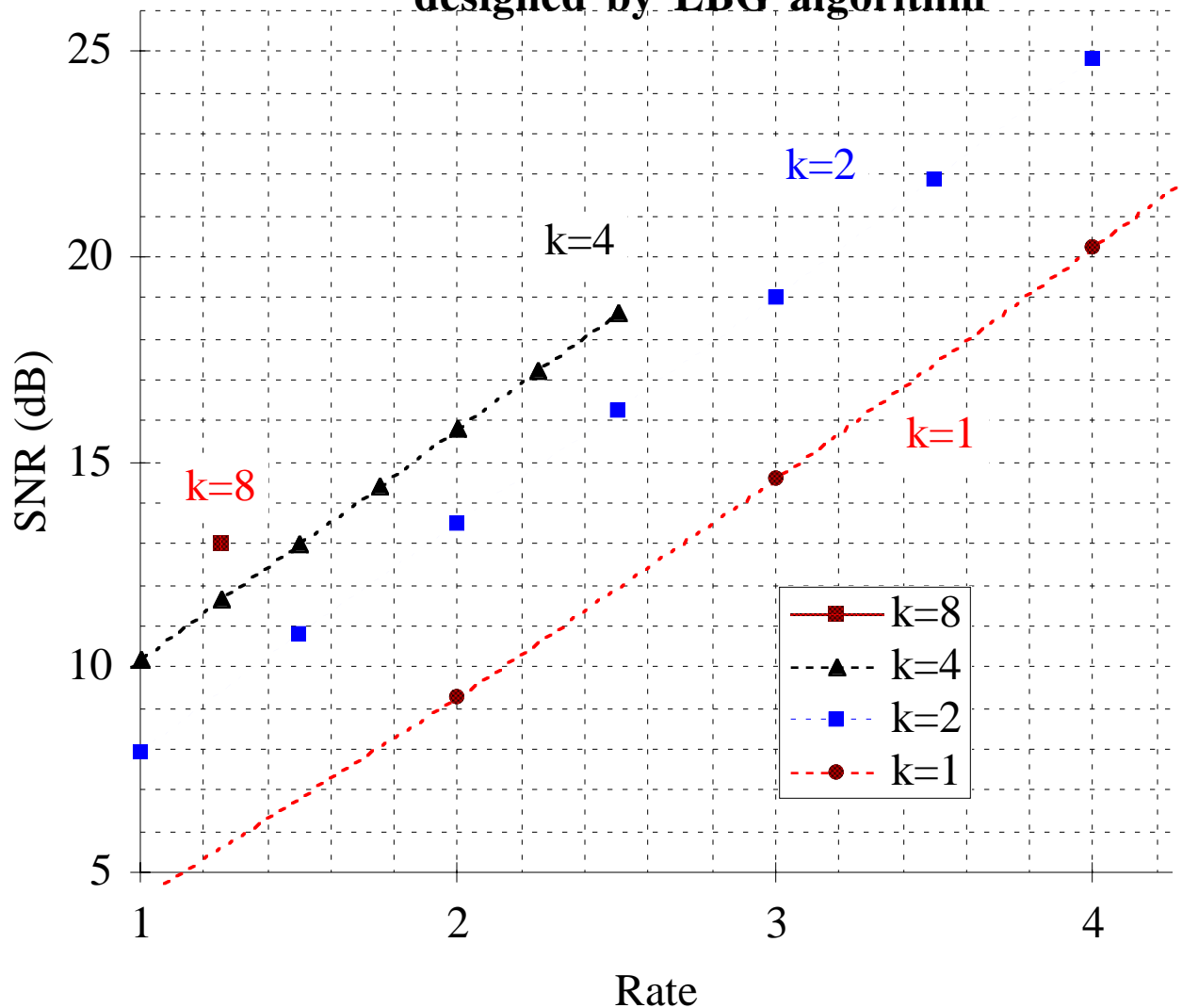


k-dimensional VQ's designed using LBG algorithm on training sequence of 640,000 samples of speech taken at 6.5 kHz sampling rate.

Notice the gains due to 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.

VQ's designed for Gaussian AR source with $a = .9$ designed by LBG algorithm



Note the gains of VQ over scalar Q:

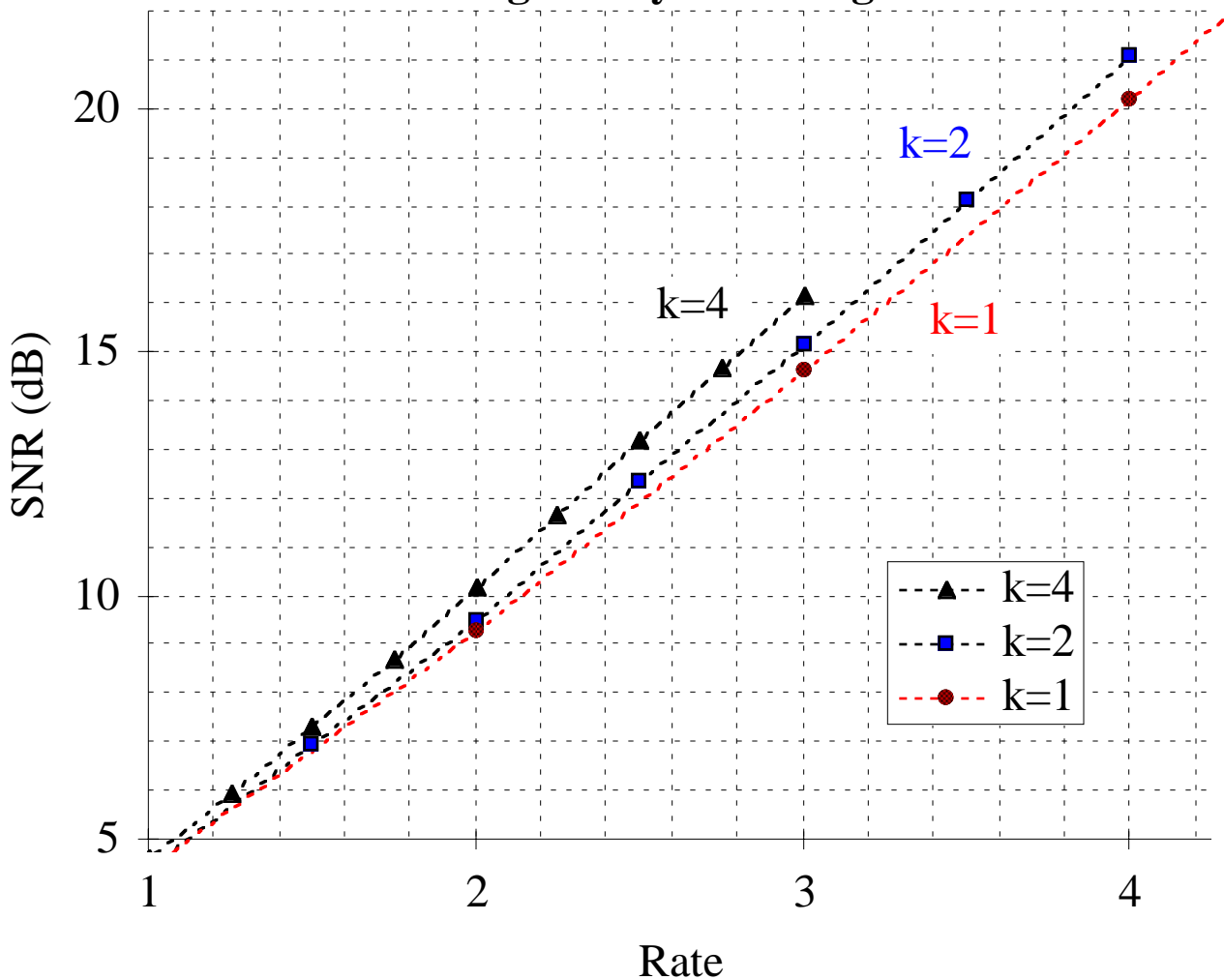
k =	2	4	8
gain \cong	4.2	6.5	7.8 dB

Compare to maximum DPCM or transform coding gain

$$10 \log_{10}(1/(1-a^2)) = 7.2 \text{ dB}$$

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

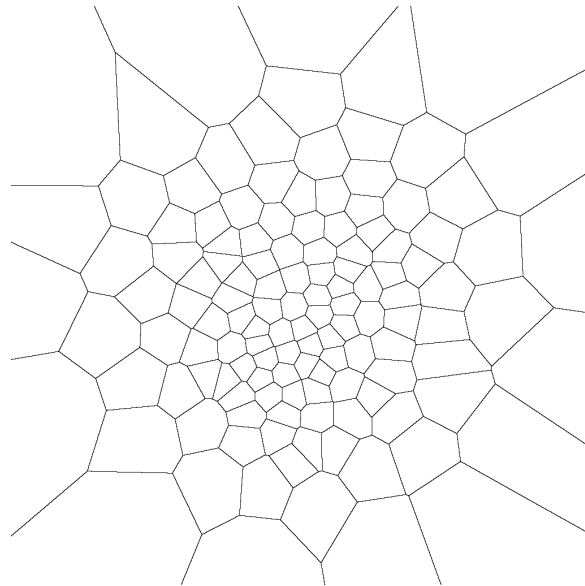
VQ's for IID Gaussian source designed by LBG algorithm



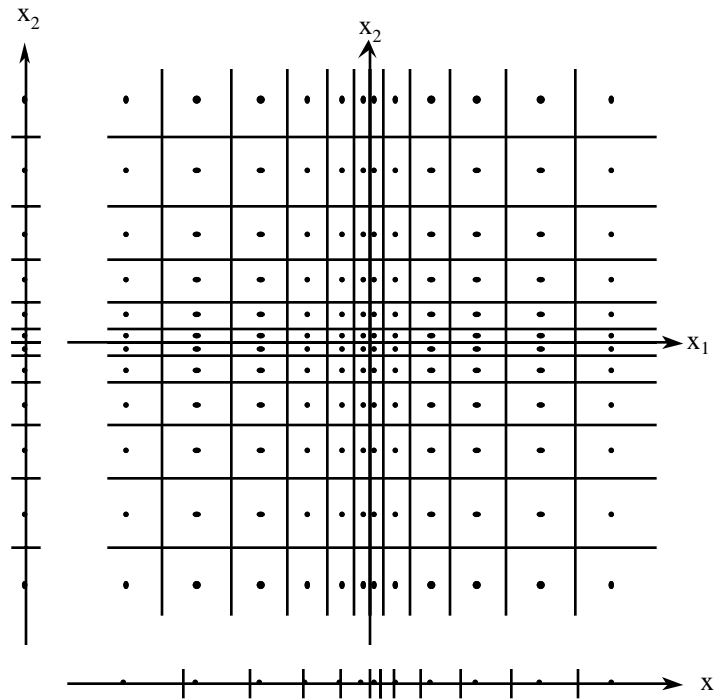
Notice the gains of VQ over scalar quantization (at rate 3, $k=4$ gains 1.6 dB), and consider the fact that neither DPCM nor transform coding give better performance than scalar quantization on this source.

Why does VQ gain over scalar quantization?

Consider an optimal 2-dimensional VQ



Compare to an optimal scalar quantizer used twice



What are the qualitative differences?

Historical Note:

It wasn't until the late 1970's that the VQ was seriously proposed and the LBG algorithm was developed. Before that, information theorists 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), and because they could easily see the large complexity associated with even moderately large dimensions. The LBG algorithm is also useful in pattern recognition where it used as a method of 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 $C = \{\underline{w}_1, \dots, \underline{w}_M\}$, $S = \{S_1, \dots, S_M\}$, $P_i = \Pr(\underline{X} \in S_i)$ and $\underline{Y} = Q(\underline{X}) =$ column vector.

If C satisfies then centroid property ($\underline{w}_i = E[\underline{X} | \underline{X} \in S_i]$), 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. $E \|\underline{Y}\|^2 = E \|\underline{X}\|^2 - E \|\underline{X} - \underline{Y}\|^2$

Proofs: Assume C satisfies centroid property ($\underline{w}_i = E[\underline{X}|\underline{X} \in S_i]$).

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

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

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

$$\begin{aligned} \text{Pf: } E X_m Y_n &= \sum_{i=1}^k \Pr(\underline{X} \in S_i) E[X_m Y_n | \underline{X} \in S_i] \\ &= \sum_{i=1}^k \Pr(\underline{X} \in S_i) E[X_m \underline{w}_{i,n} | \underline{X} \in S_i] = \sum_{i=1}^k \Pr(\underline{X} \in S_i) w_{i,n} w_{i,m} \end{aligned}$$

$$\begin{aligned} E Y_m Y_n &= \sum_{i=1}^k \Pr(\underline{X} \in S_i) E[Y_m Y_n | \underline{X} \in S_i] \\ &= \sum_{i=1}^k \Pr(\underline{X} \in S_i) w_{i,n} w_{i,m} = E X_m Y_n \end{aligned}$$

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

$$\begin{aligned} \text{Pf: } E \underline{X}^t \underline{Y} &= \sum_{i=1}^k E X_i Y_i = \sum_{i=1}^k E Y_i Y_i = E \sum_{i=1}^k Y_i^2 = E \|\underline{Y}\|^2 \\ &\quad (\text{by 2}) \end{aligned}$$

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

$$\begin{aligned} \text{Pf: } E Y_m (X_n - Y_n) &= E Y_m X_n - E Y_m Y_n = E Y_m Y_n - E Y_m Y_n = 0 \\ &\quad (\text{by 2}) \end{aligned}$$

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

$$\begin{aligned} \text{Pf: } E \underline{Y}^t (\underline{X} - \underline{Y}) &= E \underline{Y}^t \underline{X} - E \underline{Y}^t \underline{Y} = E \|\underline{Y}\|^2 - E \|\underline{Y}\|^2 = 0 \\ &\quad (\text{by 3}) \end{aligned}$$

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

$$\begin{aligned} \text{Pf: } E \|\underline{X} - \underline{Y}\|^2 &= E \|\underline{X}\|^2 - 2 E \underline{X}^t \underline{Y} + E \|\underline{Y}\|^2 \\ &= E \|\underline{X}\|^2 - 2 E \|\underline{Y}\|^2 + E \|\underline{Y}\|^2 = E \|\underline{X}\|^2 - E \|\underline{Y}\|^2 \\ &\quad (\text{by 3}) \end{aligned}$$

Generalizations to other distortion measures

voronoi regions?

centroids?

design algorithms

try magnitude error

try rth power of Euclidean distance

Voronoi regions for "difference" distortion measures: $f(\|\underline{x}-\underline{y}\|)$,
f monotonic

Centroid still good for weighted MSE?

for sum of magnitude error, get taxi density and funny
quantization cells.

Types

- Fixed-rate vs. variable-rate
(Based on whether encoder is fixed or variable rate)
- Unstructured vs. structured
There are many types of structured VQ, e.g. transform coding.

Block diagrams

