**Introduction**

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

We assume source samples are real-valued.

The encoder receives the sequence of source samples X₁, X₂, ... and produces a sequence of bits Z₁,Z₂,..., 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**: 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
    
    \[
    Z₁, Z₂, ..., Zₖ = α(X₁, ..., Xₖ), \quad Zₖ₊₁, Zₗ = α(Xₖ₊₁, ..., Xₗ), \quad ...
    \]

  - 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
    
    \[
    Y₁, ..., Yₖ = β(Z₁, ..., Zₖ), \quad Yₖ₊₁, ..., Y₂ₖ = β(Zₖ₊₁, ..., Z₂ₖ), \quad ...
    \]

1 Also known as a *block code*.

- 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, $Y_1, \ldots, Y_k$ cannot always equal $X_1, \ldots, X_k$.

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 blocks of 64 pixels.

**ADDITIONAL CHARACTERISTICS OF A VQ**

The *binary codebook* $C_b$:

$$C_b = \{ \alpha(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^k \}, \quad (\mathbf{x} \text{ is shorthand for } (x_1, \ldots, x_k))$$

$$= [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_M] \subset \{0,1\}^L$$

(an ordered set)

where $\mathbf{c}_i = (c_{i,1}, \ldots, c_{i,L}) \in \{0,1\}^L$ is the $i$th 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* $C$:

$$C = [\beta(\mathbf{c}_1), \beta(\mathbf{c}_2), \ldots, \beta(\mathbf{c}_M)]$$

(an ordered set)

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

where $\mathbf{w}_i = (w_{i,1}, \ldots, w_{i,k}) \in \mathbb{R}^k$ is the $i$th codevector.

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

Note the convention:

$$\beta(\mathbf{c}_i) = \mathbf{w}_i$$
• The quantization rule or reproduction rule $Q$:
  \[ Q(x) = \beta(\alpha(x)) \]
  This summarizes the encoding and decoding from $X$ to $Y$. But it loses track of the bits produced by the encoder.

• The quantizing/encoding partition $S$:
  \[ S = [S_1, S_2, \ldots, S_M] \]
  (an ordered set)
  where $M$ is called the size of the partition and the VQ, and
  \[ S_i = \{ x \in \mathbb{R}^k : \alpha(x) = c_i \} = \{ x \in \mathbb{R}^k : Q(x) = w_i \} \]
  is the $i$th quantizing/encoding cell

Definition: A partition of $\mathbb{R}^k$ is a collection of disjoint subsets of $\mathbb{R}^k$ whose union is $\mathbb{R}^k$. The elements of the collection are called cells.

It follows that the encoding rule is
\[ \alpha(x) = c_i \text{ when } x \in S_i \]
and the quantization rule is
\[ Q(x) = w_i \text{ when } x \in S_i \]

MORE DETAILED BLOCK DIAGRAM OF A VQ

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

source vector $\mathbf{X}$
\[ \xrightarrow{\alpha_1} \text{partition } S \]
index $I$
\[ \xrightarrow{\alpha_b} \text{binary codeword } \mathbf{Z} = c_1 \]

codevector $\mathbf{Y} = w_i$
\[ \xrightarrow{\beta_1} \text{codebook } \mathbf{C} \]
index $I$
\[ \xrightarrow{\beta_b} \text{binary decoder } \mathbf{Z} = c_1 \]

Encoder: The first block implements the partition: It finds the quantizing cell in which $x$ lies, and outputs its index $I$, i.e. $I = \alpha_1(x)$, where
\[ \alpha_1(x) = i \text{ if } x \in S_i \]
The second block is the binary encoder, which assigns the index its binary codeword, i.e. $Z = \alpha_b(I)$, where
\[ \alpha_b(i) = c_i \]
$\alpha_1$ is called the index encoding rule or partition rule$^2$
$\alpha_b$ is called the binary encoding rule. It is one-to-one.
\[ \alpha(x) = \alpha_b(\alpha_1(x)) \]

$^2$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. $I = \beta_b(Z)$, where

$$\beta_b(z) = i \text{ when } z = c_i$$

The second block produces the codevector indexed by $I$, i.e. $Y = \beta(I)$,

$$\beta_I(i) = w_i$$

$\beta_I$ is called the *binary decoding rule*

$\beta_I$ is called the *codebook rule*

$$\beta(z) = \beta_I(\beta_b(z))$$

---

**SUMMARY:** A VQ is CHARACTERIZED BY

Dimension: $k$

Codeword length: $L$

Size: $M$

Encoding rule: $\alpha$

characterized by

- **Partition:** $S = [S_1, S_2, \ldots, S_M]$
- **Binary codebook:** $C_b = [c_1, c_2, \ldots, c_M]$
- **Encoding rules:**
  $$\alpha(x) = c_i \text{ when } x \in S_i, \quad \alpha(I) = i \text{ when } x \in S_i, \quad \alpha_b(i) = c_i$$

Decoding rule: $\beta$

characterized by

- **Codebook:** $C = [w_1, w_2, \ldots, w_M]$
- **Decoding rules:**
  $$\beta(z) = w_i \text{ when } z = c_i, \quad \beta_b(z) = i \text{ when } z = c_i, \quad \beta_I(i) = w_i$$

Quantization rule: $Q$ (determined by $S$ and $C$)

$$Q(x) = \beta(\alpha(x)) = w_i \text{ when } x \in S_i$$

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)
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

\[ R \approx \frac{\log_2 M}{k} \]

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

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

\[
D = D_X(Q) = D_X(S, C) \quad \text{(the subscript } x \text{ will often be omitted)}
\]

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

\[
= \frac{1}{k} E \|\mathbf{X}-\mathbf{Y}\|^2 = \frac{1}{k} E \|\mathbf{X}-Q(\mathbf{X})\|^2 = \frac{1}{k} \int \|\mathbf{x}-Q(\mathbf{x})\|^2 f_X(\mathbf{x}) \, d\mathbf{x}
\]

\[
= \frac{1}{k} \sum_{i=1}^{M} \int_{S_i} \|\mathbf{x}-\mathbf{w}_i\|^2 f_X(\mathbf{x}) \, d\mathbf{x}
\]

where

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

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

**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

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

where \( n >> k \) is the length of the data.

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

\[
D_{\text{emp}} \approx D
\]

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: \( k=1, \ M=4, \ C_1 = \{ w_1, w_2, w_3, w_4 \} = \{ -4, -1, 1, 4 \}, \ R = 4 \text{ bits/samp} \)

The scalar quantizer applied to the source sequence:

A Vector Quantizer: \( k = 2, \ M = 10, \ R = \frac{1}{2} \log_2 10 = 1.66 \text{ bits/sample} \)

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

The idea behind this 2-dimensional VQ is that with the scalar quantizer, successive source samples are mostly quantized into identical or adjacent quantization "levels". This suggests that into the codebook \( C_2 \) we put only pairs of levels that are identical or adjacent. In comparison, the scalar quantizer permits any of the 16 possible pairs of successive levels in the set

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

The fact that 10 is less than 16 is the reason why the vector quantizer has lower rate than the scalar quantizer.
The VQ applied to the source sequence:

```
  1
  4
```

the pairs of circles connected by lines indicate the chosen codevectors

The distortion and rate for the scalar and vector quantizers:

```
D
  k=2, M=10
  k=1, M=4
```

```
R
  1
  2
```

the distortions shown are not actual MSE, just representative values

Another vector quantizer: \( k = 2, \ M = 4, \ R = \frac{1}{2} \log_2 4 = 1 \ \text{bit/sample} \)

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

The codevectors and corresponding quantization cells are shown below.

This VQ applied to the source sequence:
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 overload region $R_o$, consisting of all $x$ such that $\|x-Q(x)\|^2 >> D$
- The granular region $R_G$, consisting of all $x$ such that $\|x-Q(x)\|^2 \lesssim D$

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

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

$$D = D_G + D_O$$

where

$$D_G = \text{granular distortion} = \int_{R_G} \|x-Q(x)\|^2 f_X(x) \, dx$$

$$D_O = \text{granular distortion} = \int_{R_G} \|x-Q(x)\|^2 f_X(x) \, dx$$

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. $E \|Y\|^2 = E \|X\|^2 - E \|X-Y\|^2$
- 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
- Variable-rate VQ -- optimality properties and high-resolution theory.
VQ 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 \( X = (X_1, \ldots, X_k) \). 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 \( X \) 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) \( f_X(x) \).

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 $X$, $k$, $M$*, if it has the smallest MSE of any quantizer with dimension $k$ and size $M$ applied to the random vector $X$.

Facts:
- For any $X$, $k$, $M$, 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:
$$\delta_X(k,M) = \text{least MSE of any quantizer with size } M \text{ for } X = (X_1,...,X_k).$$
$$\delta_X(k,R) = \text{least MSE of any quantizer with rate } R \text{ for } X = (X_1,...,X_k).$$
These are called OPTA³ functions (Optimum Performance Theoretically Attainable).

³It'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 $C = \{w_1, \ldots, w_M\}$ and a random vector $X$, 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

$$D(S,C) = \frac{1}{k} \int ||x-Q(x)||^2 f_X(x) \, dx$$

where $Q(x) = w_i$ when $x \in S_i$. 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 $x$, to choose $Q(x)$ to be a codevector $w_i$ such that $||x-w_i|| \leq ||x-w_j||$ for all $j \neq i$. This means that if $x$ is closer to $w_i$ than to any other $w_j$, then $x$ should be placed in cell $S_i$. In other words, the cell $S_i$ should contain the set

$$V_i = \{ x : ||x-w_i|| < ||x-w_j||, \text{ for all } j \neq i \}.$$

If $x$ is equally closest to two or more of the codevectors, then we can place $x$ in the cell of any one of the closest codevectors.
On the other hand, the set $W$ of all $x$'s that are equally closest to two or more codevectors is a $(k-1)$-dimensional subset of $k$-dimensional space, and since $X$ 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 $Q(x)$ on a set of probability zero, it doesn't actually matter how these $x$'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 $w_i$ 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 $C = \{w_1, \ldots, w_M\}$, let

$$V_i = \{x : ||x-w_i|| < ||x-w_j||, \text{ for all } j \neq i\}.$$  

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

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

where $A \cong B$ means $\Pr(X \in (A-B) \cup (B-A)) = 0$.

Notes:

- It is interesting to notice that an optimal partition can be chosen without regard to the probability distribution of $X$.

- An interesting viewpoint: 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 produce the codevector that is closest in Euclidean distance to the source vector $x$. 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 $X$. Specifically, if the source produces $x$, the decoder should produce $y$ as close as possible to $x$ and "how close" should not depend on $f_X(x)$.

- A partition $S$ such that $S_i \cong V_i$ 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 $w_i$ as well to some other point. That is, $T_i$ is a subset of

$$\{x : ||x-w_i|| = ||x-w_j|| \text{ some } j, \text{ and } ||x-w_i|| \leq ||x-w_j|| \text{ for all } j\}$$

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"

\[ S_i = \{ x : \| x - w_i \| < \| x - w_j \|, \text{ for all } j \neq i \} \]

or

\[ S_i = \{ x : \| x - w_i \| \leq \| x - w_j \|, \text{ for all } j \neq i \}. \]

• To find the Voronoi partition, draw perpendicular bisectors between each pair of codevectors. These are (k-1)-dimen'l hyperplanes, each dividing \( \mathbb{R}^k \) into two half spaces. The Voronoi region \( S_i \) is the intersection of the halfspaces containing \( w_i \).

• 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 collinear, their three perpendicular bisectors meet at a point.
Suppose we are given a partition \( S = \{S_1, \ldots, S_M\} \) and a random vector \( X \), and suppose we seek the best codebook \( C = \{w_1, \ldots, w_M\} \) to use with this partition. That is, we seek the codebook \( C \) that minimizes the distortion \( D_X(S, C) \). The answer comes immediately when one considers that the decoder is, in effect, told which quantization cell \( S_i \) the source vector \( X \) resides and must produce a value \( Y \) that minimizes \( E \|X - \hat{X}\|^2 \). This value \( Y \) may be considered to be an estimate of \( X \). In this light, the role of the decoder is to estimate \( X \) from knowledge of the cell in which \( X \) resides. From conventional minimum MSE estimation theory, we know that when the decoder is told \( X \in S_i \), it should produce the conditional expectation \( E[X|X \in S_i] \). That is, the optimal codevector for the cell \( S_i \) is \( E[X|X \in S_i] \).

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

\[
D_X(S, C) = \sum_{i=1}^{M} Pr(X \in S_i) \int_{S_i} \|X - w_i\|^2 \frac{f_X(x)}{Pr(X \in S_i)} \, dx = \sum_{i=1}^{M} Pr(X \in S_i) \int_{S_i} \|X - w_i\|^2 f_X(x|X \in S_i) \, dx
\]

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

Optimality Condition 2: Optimal codebook for a given partition.

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

\[
w_i = E[X|X \in S_i] = \int_{S_i} x f_X(x|X \in S_i) \, dx, \quad i = 1, \ldots, M
\]

where

\[
f_X(x|X \in S_i) = \begin{cases} \frac{f_X(x)}{Pr(X \in S_i)}, & x \in S_i \\ 0, & \text{else} \end{cases}
\]

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 Conditions:

**Corollary:** Optimal partition and codebook.

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

**Notes:**
- 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 satisfies 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 \( w_i \) by \( w_i + \varepsilon z \) where \( z \) is an arbitrary vector and \( \varepsilon \) is some scalar. If the VQ is locally optimal, then for any \( z \) there is an \( \varepsilon_0 \) such that for all \( \varepsilon \leq \varepsilon_0 \), 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 \( \varepsilon \). Then there must exist some \( \varepsilon_0 \) 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 \( \varepsilon_0 \) such that for all \( \varepsilon \leq \varepsilon_0 \), the perturbed VQ has the same or larger 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 \( C = \{ w_1, \ldots, w_M \} \).

1. Given \( x \) compute \( ||x - w_i||^2 \) for each \( i \)
2. Find the \( i \) that minimizes \( ||x - w_i||^2 \) and send \( c_i \)

Table-Lookup Decoding

Store the codebook \( C = \{ w_1, \ldots, w_M \} \) in a table.

When the decoder is given \( Z = c_i \), it outputs \( Y = w_i \) as the reproduction of \( X \).

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

<table>
<thead>
<tr>
<th>storage</th>
<th>ops/sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>encoder</td>
<td>Mkb</td>
</tr>
<tr>
<td>decoder</td>
<td>Mkb</td>
</tr>
</tbody>
</table>

\(b = \text{no. of bits/component} \approx R + 3\) to 5 is usually sufficient

**encoding operations:**

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

**The "curse of dimensionality"**

Since \(M=2^{kR}\), both storage and computation increase exponentially with \(k\) and \(R\).

The dimension-rate product \(kR\) is the determining factor. For example, e.g. 
\(k=64\) and \(R=1\), then \(M=2^{64} = 1.8 \times 10^{19}\).

This is a critical limitation. Generally, in practice 
\(kR \leq 10\) or 12.

While modern computers can implement 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.