

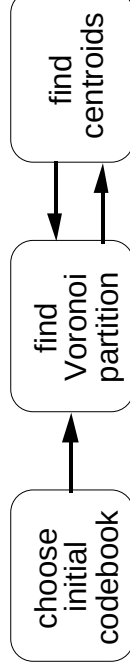
GENERALIZED LLOYD VQ DESIGN ALGORITHMS

These are iterative algorithms in the spirit of Lloyd's algorithm⁴ 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

⁴Stephen Lloyd, unpublished Bell Labs memorandum July 1957.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 $C^{(i)} = \{\underline{w}_1^{(i)}, \dots, \underline{w}_M^{(i)}\}$ produced by the i th iteration,

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

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

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

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

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

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

3. Stopping conditions

Possibilities include:

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

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

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.

49

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 Gray⁵. The algorithm was also introduced in the pattern recognition literature for a somewhat different purpose as the "k-means algorithm".

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

50

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, $\underline{t}_1, \underline{t}_2, \dots, \underline{t}_N$, 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 $C^{(0)} = \{\underline{w}_1^{(0)}, \dots, \underline{w}_M^{(0)}\}$.
- Iterate the following steps until the centroids change little and/or the distortion changes little.

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

1. Partition training set: $\tilde{S}^{(i+1)} = \{\tilde{S}_1^{(i+1)}, \dots, \tilde{S}_M^{(i+1)}\}$
 $\tilde{S}_j^{(i+1)} = \{\underline{t}_i : \underline{t}_i \text{ closer to } \tilde{\underline{w}}_j^{(i)} \text{ than to } \tilde{\underline{w}}_{j'}^{(i)}, j' \neq j\}$

2. Find empirical centroids:

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

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

51

In Step 1, for each training vector, we apply full search encoding to find 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 $\tilde{C}^{(i)} = \{\tilde{\underline{w}}_1^{(i)}, \dots, \tilde{\underline{w}}_M^{(i)}\}$ produced by the i th iteration,

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

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

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

52

Issues:

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

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

$$D_{\text{train}} = \frac{1}{N} \sum_{j=1}^N \|t_j - Q(t_j)\|^2$$

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 (*) and (**).

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.

53

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 $t_1, t_2, \dots, t_N$, and that we desire to choose codevectors $w_1, \dots, w_M$ to minimize the training distortion

$$D_{\text{train}} = \sum_{j=1}^N \min_j \|t_j - w_j\|^2$$

To attempt to do this, repeatedly perform the following:

$$\tilde{w}_j^{(i+1)} = \frac{1}{N_j^{(i)}} \sum_{j: t_j \text{ closest to } \tilde{w}_j^{(i)}} t_j$$

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

Performing the above will not increase and usually decrease

$$\sum_{j=1}^N \min_j \|t_j - w_j\|^2$$

54

7. Training distortion vs. actual distortion:

Suppose that the LBG algorithm is run on a training sequence $t_1, t_2, \dots, t_N$ 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,

$$D_{\text{train}} = \frac{1}{N} \sum_{j=1}^N ||t_j - Q(t_j)||^2$$

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.

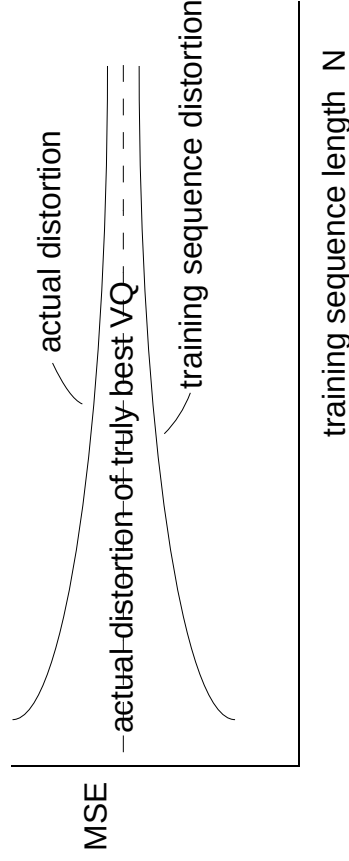
Because training distortion can be substantially less than actual distortion, it is customary to estimate the actual distortion by running the VQ on a *test sequence* that is distinct from the training sequence.

55

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 $50M$, 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.

56

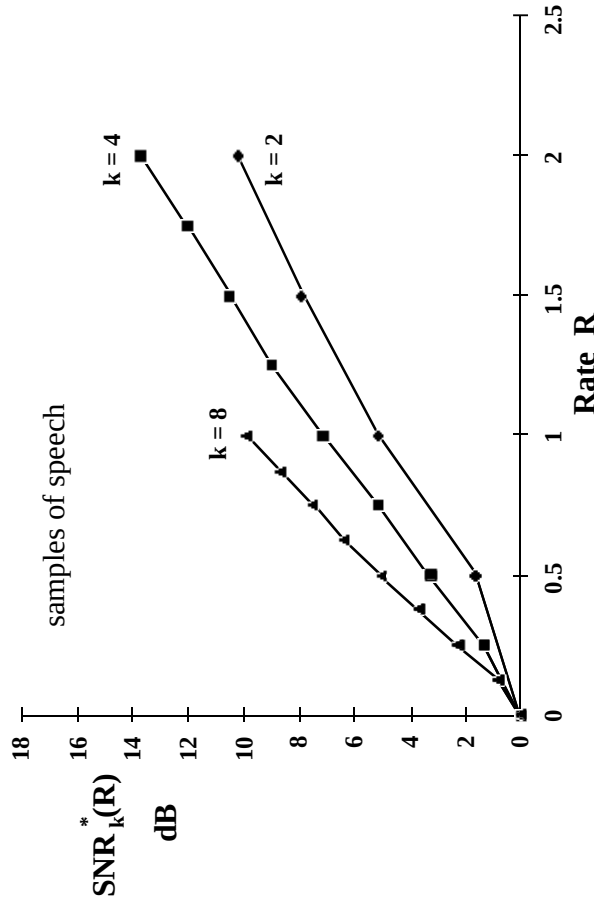
8. Complexity of VQ design

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

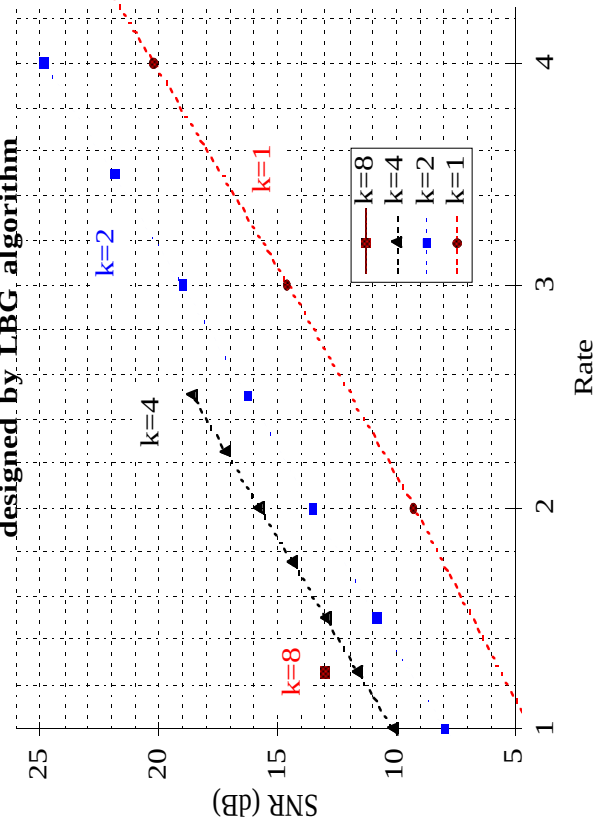


k -dimensional VQ's designed using the LBG algorithm on a training sequence of 640,000 samples of speech taken at 6.5 kHz sampling rate.

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.

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



Gains (in dB) of VQ over scalar Q:

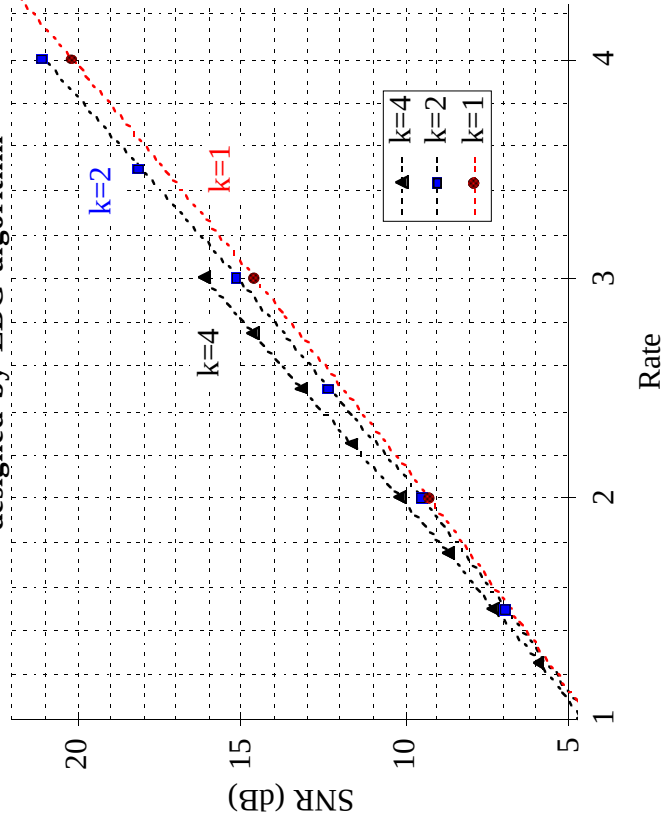
k	=	2	4	8
gain	≡	4.2	6.5	7.8

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?

VQ's for IID Gaussian source
designed by LBG algorithm

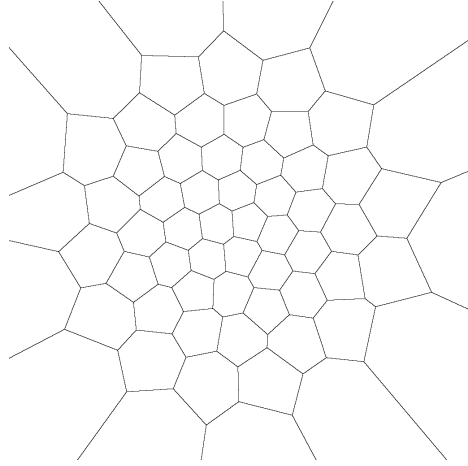


Again notice the gains of VQ over scalar Q (at rate 3, $k=4$, VQ gains 1.6 dB)
Is it intuitive that VQ gains over scalar Q, even for an IID source?
High-resolution theory will show why.

61

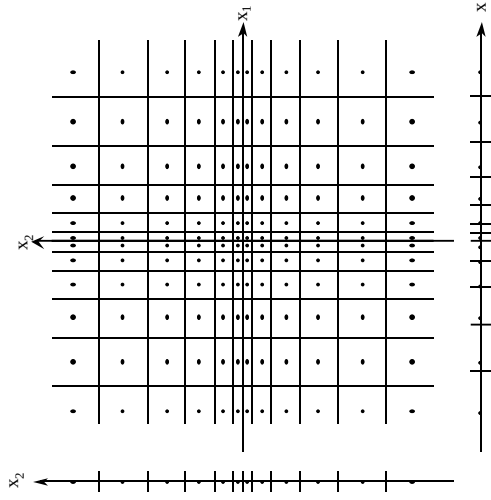
COMPARISON OF QUANTIZERS FOR AN IID GAUSSIAN SOURCE

An LBG designed 2-dimensional VQ



$M = 64, R = 3$

An optimal scalar quantizer used twice



$M_{\text{scalar}} = 12, M = 144, R = 7.17$

How are the partitions and codebooks different?

62

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 sizeable 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

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

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. $E \underline{X}_m \underline{Y}_n = E \underline{Y}_m \underline{Y}_n$
3. $E \underline{X}^t \underline{Y} = E \|\underline{Y}\|^2$
4. $E \underline{Y}_m (\underline{X}_n - \underline{Y}_n) = 0$
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 $0 \leq \alpha \leq 1$.

Proof of Fact A:

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

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

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

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

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

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

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

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

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

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

65

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

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

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

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

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

Follows from the definition of a convex cell.

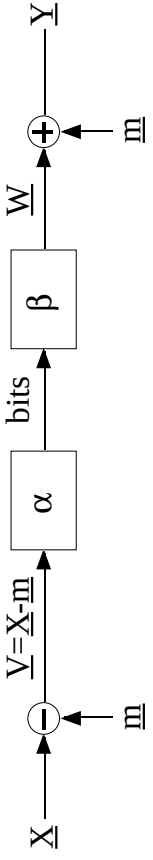
Notes:

- 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.

66

1. Shifting

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



This shifted quantizer is characterized by $k, M, L, S', C', Q', C_b, \alpha', \beta'$:

$$S' = \{S'_1, \dots, S'_M\} = S' + \underline{m}, \quad S'_i = S_i + \underline{m} = \{X + \underline{m} : X \in S_i\}$$

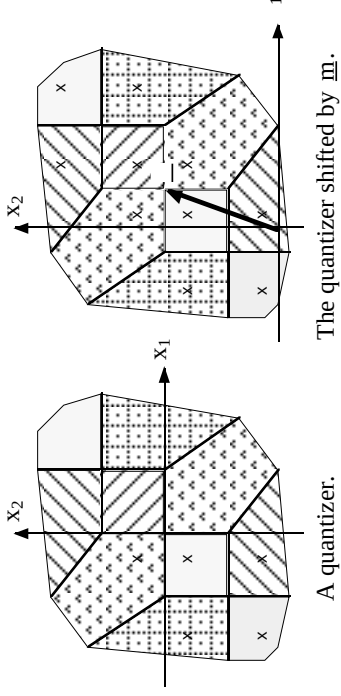
$$C' = \{\underline{w}'_1, \dots, \underline{w}'_M\} = C' + \underline{m}, \quad \underline{w}'_i = \underline{w}_i + \underline{m}$$

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

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

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

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.



Facts:

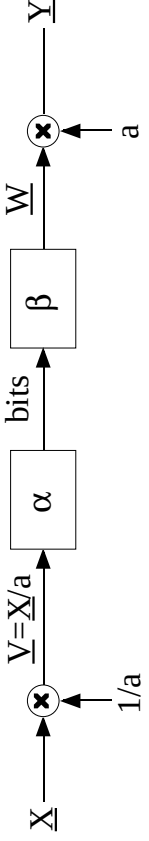
1. The rate of the shifted quantizer is the same as the rate of original quantizer.
2. The MSE of the shifted quantizer applied to $\underline{X}$ is the same as the MSE of the original quantizer applied to $\underline{Y}$.
3. If k, M, L, S, C is optimal for random vector $\underline{Y}$ with pdf $p_Y(\underline{Y})$,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

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

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 $k, M, L, S, C, Q, C_b, \alpha, \beta$, and a positive scale factor a , the *scaled* version of this quantizer is shown below:



This scaled quantizer is characterized by $k, M, L, S', C', Q', C_b, \alpha', \beta'$:

$$S' = \{S'_1, \dots, S'_M\} = aS, \quad S'_i = aS_i = \{a\mathbf{x} : \mathbf{x} \in S_i\}$$

$$C' = \{\mathbf{w}'_1, \dots, \mathbf{w}'_M\} = aC, \quad \mathbf{w}'_i = a\mathbf{w}_i$$

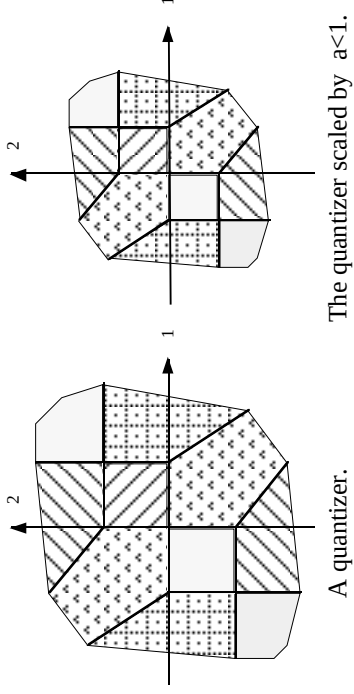
$$Q'(\mathbf{x}) = aQ(\mathbf{x}/a)$$

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

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

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

69



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 $\mathbf{x}$ is a^2 times the MSE of the original quantizer applied to $\mathbf{y}$.
3. If k, M, L, S, C is optimal for random vector $\mathbf{y}$ with pdf $p_Y(\mathbf{y})$, then the scaled quantizer k, M, L, S', C' is optimal for the random vector $\mathbf{x} = a\mathbf{y}$, with pdf $p_X(\mathbf{x}) = \frac{1}{a} p_Y(\mathbf{x}/a)$.

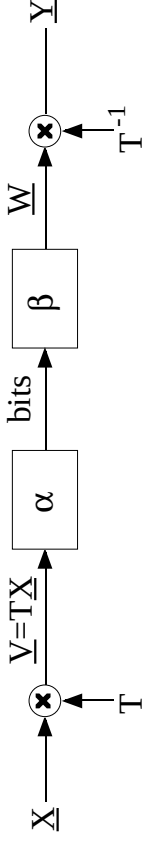
One can prove this by showing that if there were a better quantizer for $\mathbf{x}$, then one could find a better quantizer for $\mathbf{y}$, which would contradict the assumed optimality of the original quantizer for $\mathbf{y}$.

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.

70

3. Linear Transforming

This is a kind of vector generalization of scaling. Given a quantizer characterized by $k, M, L, S, C, Q, C_b, \alpha, \beta$, and a $k \times k$ orthogonal matrix⁶ T , the transformed version of this quantizer is shown below:



This transformed quantizer is characterized by $k, M, L, S', C', Q', C_b, \alpha', \beta'$:

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

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

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

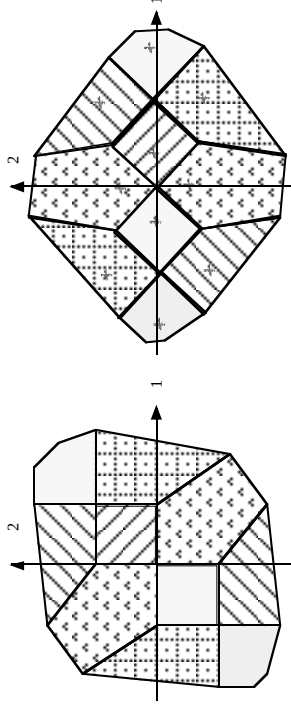
$$\alpha'(\underline{x}) = \alpha(T\underline{x})$$

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

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.



Facts:

1. The rate of the 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 $p_v(\underline{v})$, then the scaled quantizer k, M, L, S', C' is optimal for the random vector $\underline{x} = T^{-1}\underline{v}$, with pdf $p_x(\underline{x}) = p_v(T\underline{x})$.

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 make nonuniform scalar quantizers from a uniform scalar quantizer.

73

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 viewed as *time-sharing* the two.) It is characterized by

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

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

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

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

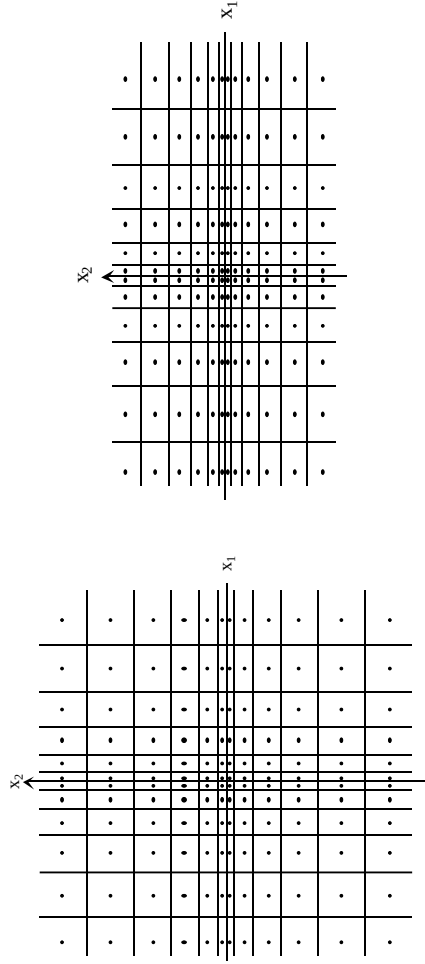
$$R = \frac{L_k + L_m}{k+m} = \frac{k}{k+m} R_k + \frac{m}{k+m} R_m$$

2. The MSE of the product quantizer is

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

74

Examples of the product of two scalar quantizers



Notes:

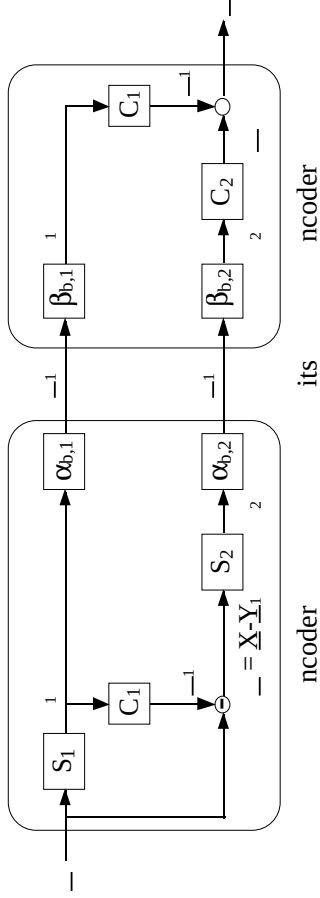
1. In the same way, one can form the product of three or more quantizers.
2. Product quantizers are not really a new VQ method. Rather it is principally a way to view lower dimensional quantizers in higher dimensions. This will make it possible to make reasonable comparisons of quantizers with different 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

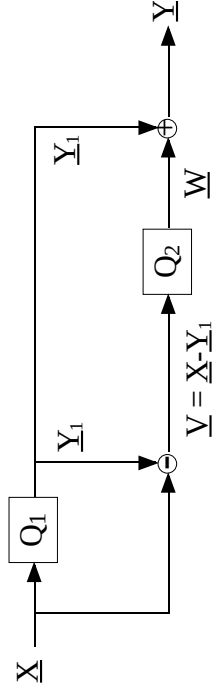
Suppose we are given two vector quantizers with the same dimension k

$k, M_1, L_1, S_1, C_1, Q_1, \alpha_1, \beta_1$ and $k, M_2, L_2, S_2, C_2, Q_2, \alpha_2, \beta_2$.

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:

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

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

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

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

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

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

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

Facts:

1. Upper bound to the rate:
 $R \leq \max\{R_1, R_2\} + \frac{1}{k}$
2. Upper bound to distortion:
 $D \leq \min\{D_1, D_2\}$

Note:

- Suppose (S_1, C_1) is a k -dimensional quantizer with rate R and mse D_1 when the source vector $\underline{X}$ has pdf $p_1(\underline{x})$, and (S_2, C_2) is a k -dimensional quantizer with rate R and mse D_2 when the source vector $\underline{X}$ has pdf $p_2(\underline{x})$. Then the union code has

$$\text{rate} \leq R + \frac{1}{k}$$

$$\text{distortion for pdf } p_1 \leq D_1$$

$$\text{distortion for pdf } p_2 \leq D_2$$

If k is large so that $\frac{1}{k}$ is small relative to R , then we see that the union code works well for both p_1 and p_2 . Previously, we'd seen only how to design a quantizer for one specific probability distribution.