There are also:

- Fast search methods for unstructured VQ coebooks.

  We will discuss some later.

  Though they can be considerably simpler, I've yet to see one whose complexity does not increase exponentially with distortion-rate product.

- Structured VQ methods

  e.g. JPEG

  We will discuss other 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.

VQ Design Algorithms

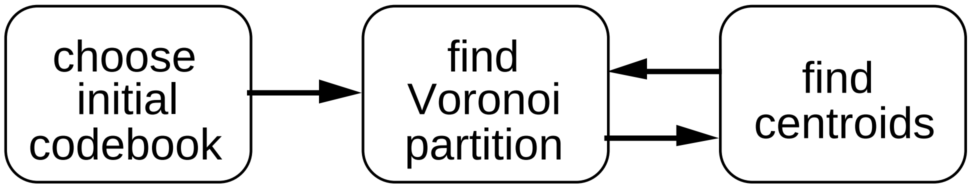
Generalized Lloyd algorithms

These are iterative algorithms, in the spirit of Lloyd's algorithm for scalar quantizer design. 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 (finding Voronoi partition or centroids) does not increase distortion, the distortion of the algorithm is guaranteed to converge.

Typically, the actual partition and codebook converge to a local optimum (i.e. a VQ that satisfies (*) and (**)), which might or might not be the globallyt optimal quantizer. But 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 suggsted 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 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.

The "design from pdf" Lloyd algorithm is seldom used

It is often used for designing scalar quantizers.

It is seldom used for designing vector quantizers (k>1) because

- the pdf of the source vector is often unknown.
- 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 that described next.

B. Design from training sequence. (LBG algorithm)

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

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

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

1. Find Voronoi "sets":

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

2. Find empirical centroids.

   $$\tilde{\underline{w}}_i = \frac{1}{N_i} \sum_{j:\underline{t}_j \in \tilde{S}_i} \underline{t}_j ,$$ where N_i = # 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 $\underline{w}_1,\ldots,\underline{w}_M$, design new centroids via

$$\tilde{\underline{w}}_i = \frac{1}{N_i} \sum_{j:\underline{t}_j \text{ closest to } \underline{w}_i} \underline{t}_j ,$$ where N_i = # train'g vect's closest to $\underline{w}_i$

Here, for each training vector $\underline{t}_j$,

- find codevector, say $\underline{w}_i$, to which it is closest, via full search.
- increment a counter that stores N_i
- add $\underline{t}_j$ to an accumator that computes the sum in $\tilde{\underline{w}}_i$.

Initial codebook:

The same choices as with Algorithm A are available.

We have a version of this algorithm available, written in C.

LBG = Linde, Buzo and Gray,
IEEE Trans. Commun., Jan. 1980

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 paritition-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 might be far from the mimimum.

The difference between training and actual distortion 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.

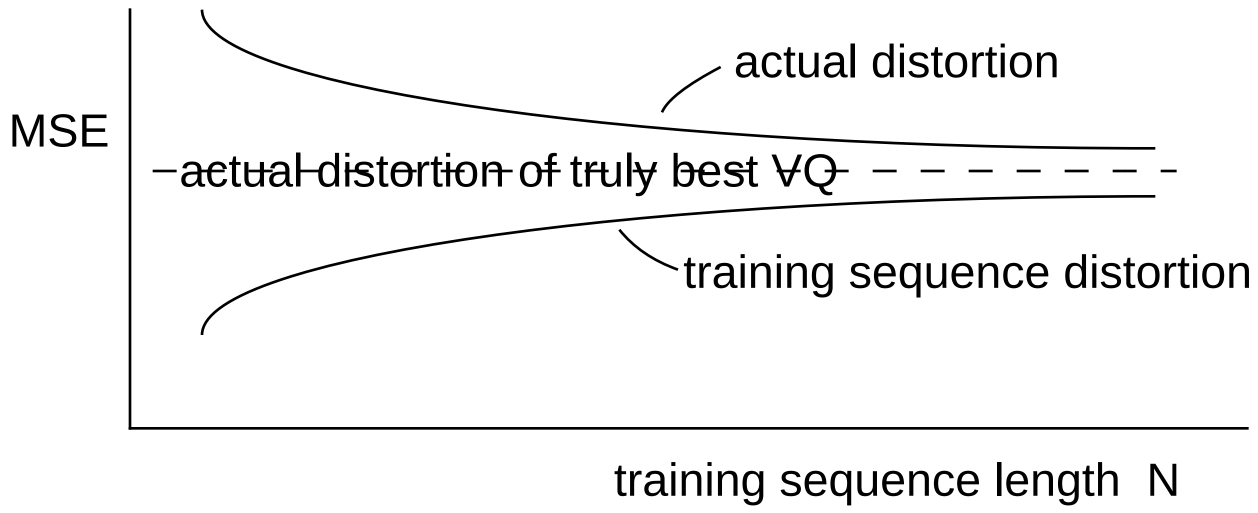
Training sequence length:

How large to make the training sequence length N?

Two issues:

- the VQ designed by the algorithm becomes better as N is made larger,
- the training distortion becomes a more accurate estimate of actual distortion.

The convergence of the actual and training distortions 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 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 distortions are reasonably close.

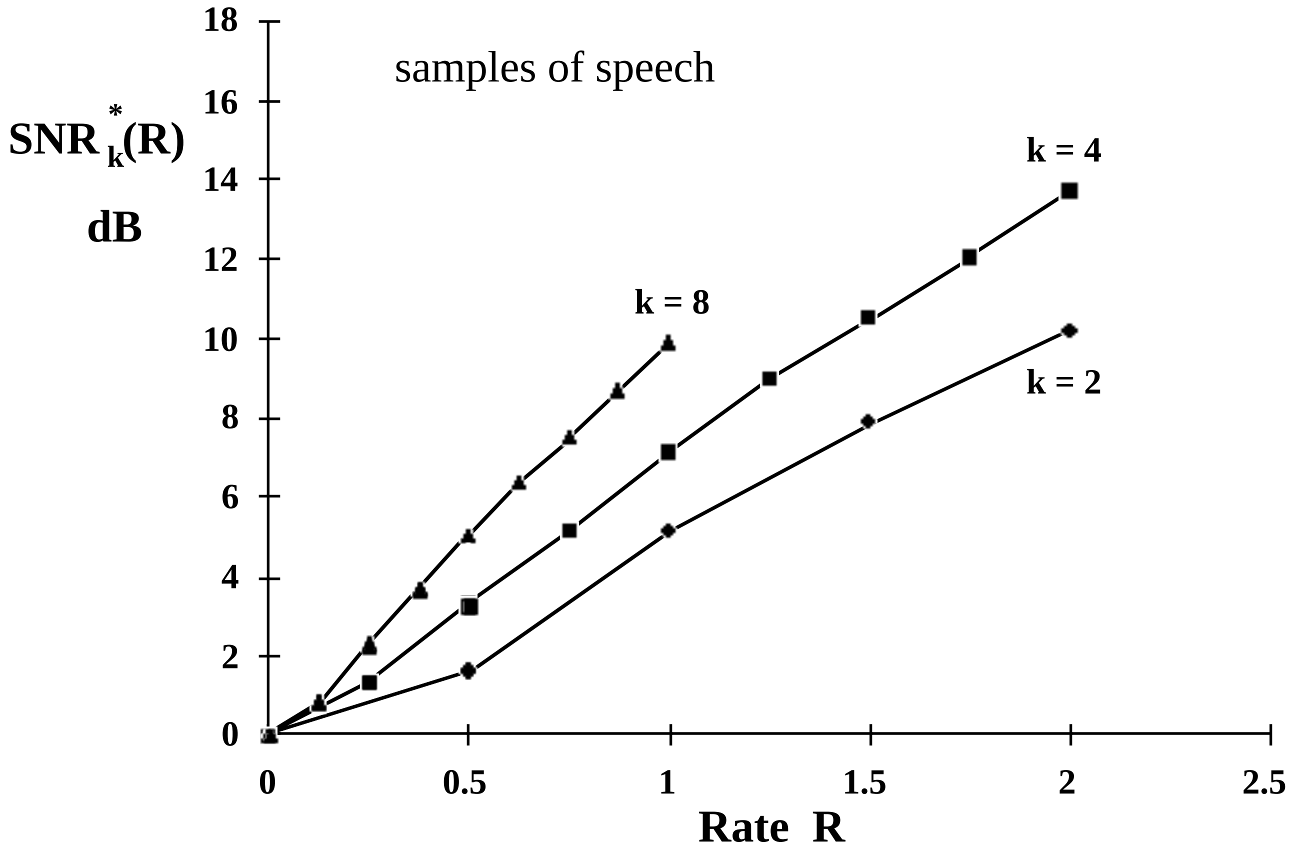
Complexity of VQ design:

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 an ustructured 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.

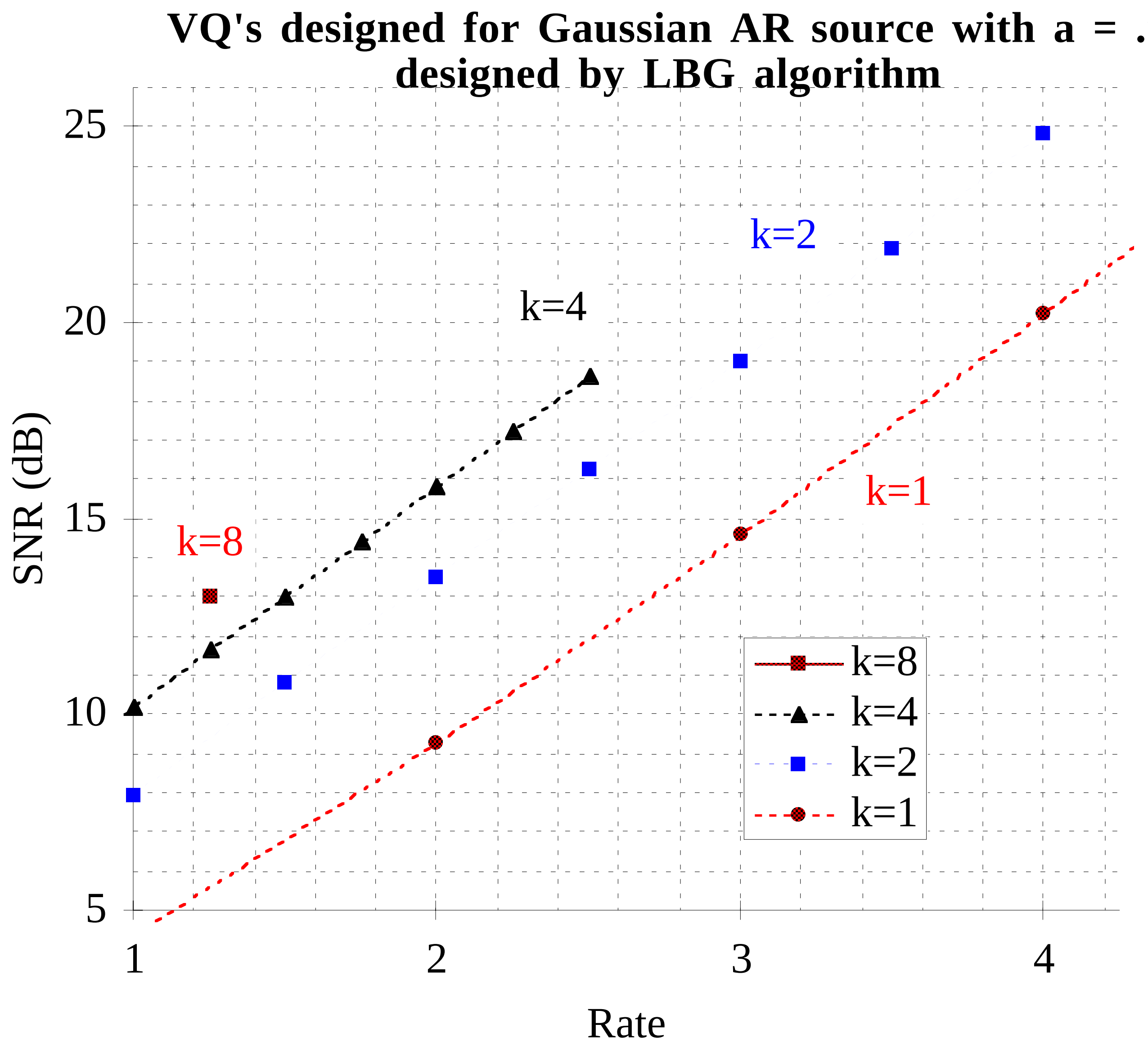
Examples of VQ's designed by the LBG algorithm



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.



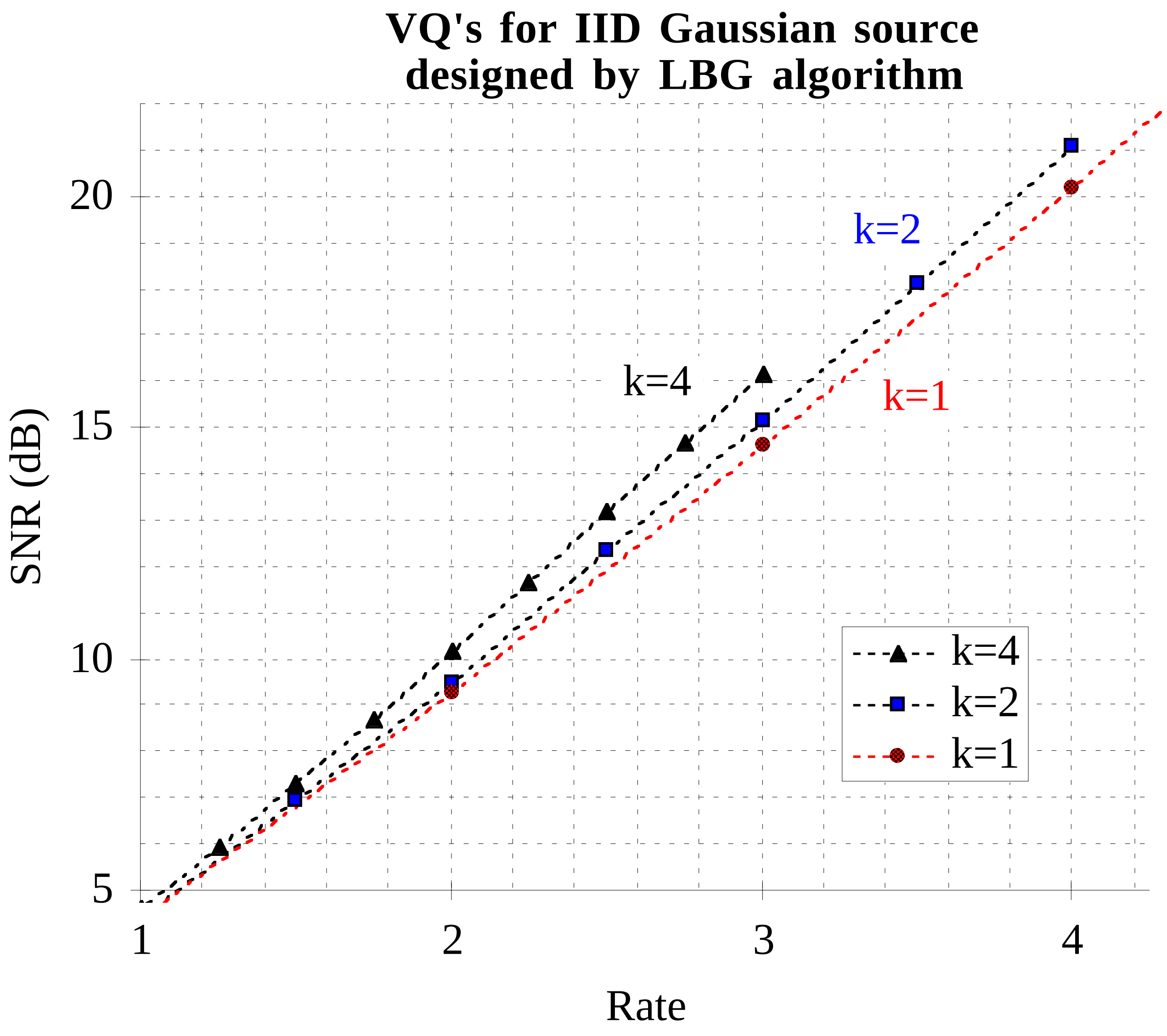
Note the gains of VQ over scalar Q:

k =	2	4	8
gain ≅	4.2	6.5	7.8 dB

Important questions:

Why does VQ gain over scalar Q?

Why does gain increase with dimension?



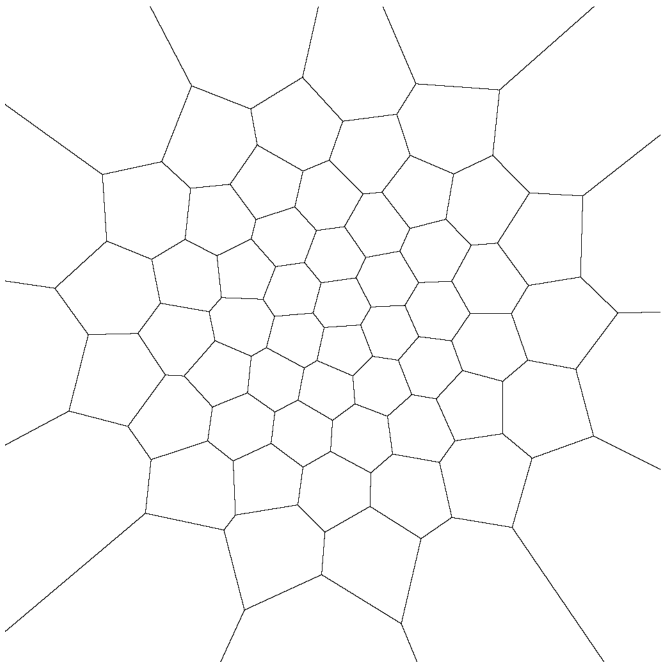
Notice the gains of VQ over scalar quantization (at rate 3, k=4 VQ gains 1.6 dB).

Important questions:

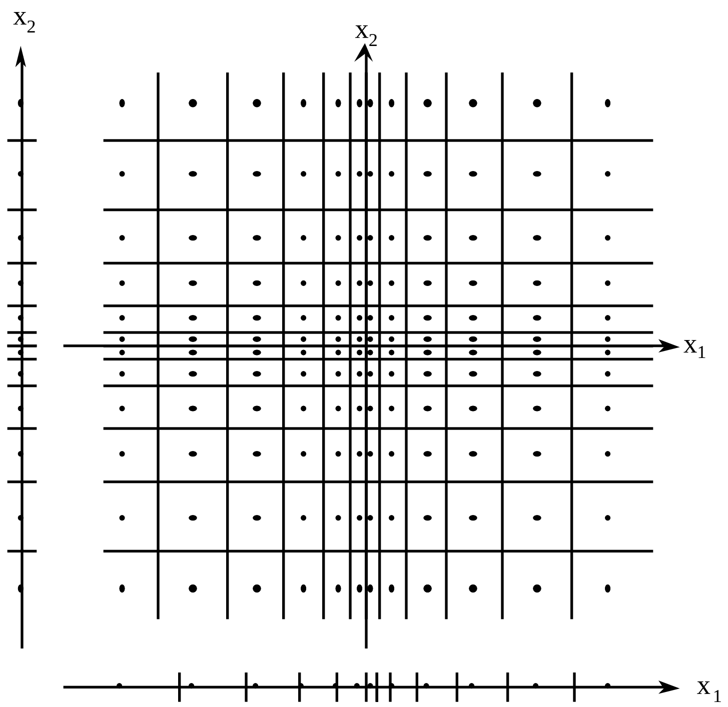
Why does VQ gain over scalar Q?

Why does gain increase with dimension?

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

Other Properties of Optimal Quantizers

Consider a VQ with

$C = \{\underline{w}_1,\ldots,\underline{w}_M\}$

$S = \{S_1,\ldots,S_M\}$,

$P_i = \Pr(\underline{X} \in S_i)$

$\underline{Y} = Q(\underline{X})$ = column vector.

If C satisfies the centroid property ($\underline{w}_i = E[\underline{X} | \underline{X} \in S_i]$) for all i, then

1. $E\,\underline{Y} = E\,\underline{X}$
2. $E\,X_m Y_n = E\,Y_m Y_n$
3. $E\,\underline{X}^t\underline{Y} = E\,||\underline{Y}||^2$
4. $E\,Y_m(X_n - Y_n) = 0$
5. $E\,\underline{Y}^t(\underline{X} - \underline{Y}) = 0$
6. $E\,||\underline{Y}||^2 = E\,||\underline{Y}||^2 - E\,||\underline{X} - \underline{Y}||^2$

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

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

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

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

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

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

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

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

Pf: $$E\, Y_m(X_n-Y_n) = E\, Y_m X_n - E\, Y_m Y_n = E\, Y_m Y_n - E\, Y_m Y_n = 0$$
(by 2)

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

Pf: $$E\,\underline{Y}^t(\underline{X}-\underline{Y}) = E\,\underline{Y}^t\underline{X} - E\,\underline{Y}^t\underline{Y} = E\,||\underline{Y}||^2 - E\,||\underline{Y}||^2 = 0$$
(by 3)

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

Pf: $$E\,||\underline{X}-\underline{Y}||^2 = E\,||\underline{X}||^2 - 2\,E\,\underline{X}^t\underline{Y} + E\,||\underline{Y}||^2$$
$$= E\,||\underline{X}||^2 - 2\,E\,||\underline{Y}||^2 + E\,||\underline{Y}||^2 = E\,||\underline{X}||^2 - E\,||\underline{Y}||^2$$
(by 3)