

Structured vector quantization

Consider several structured codes, that is, impose a structure that allows for reduced implementation complexity. Constrain codewords or codeword search.

- Lattice quantization
- Tree-structured codes
- Multistage codes
- Product codes: gain/shape codes

E.g., a uniform scalar quantizer (all bin widths equal) is a lattice quantizer. A product of k uniform quantizers is a lattice quantizer.

hexagonal lattice in 2D The generator matrix for the hexagonal lattice in \mathbb{R}^2 is

$$\begin{bmatrix} \sqrt{3} & 0 \\ 1 & 2 \end{bmatrix}$$

E_8 in 8D.

Lattice quantizers are a multidimensional generalization of uniform quantization. (Rectangular lattice corresponds to scalar uniform quantization.)

Lattice vector quantization

Codebook is subset of a regular lattice.

A lattice \mathcal{L} in \mathbb{R}^k is a set of all vectors of the form $\sum_{i=1}^n m_i u_i$, where the $\{m_i, i = 0, 1, \dots, n-1\}$, $n \leq k$ are integers and the u_i are linearly independent. We assume that $n = k$ so the *generator vectors* $\{u_i; i = 0, 1, \dots\}$ span \mathbb{R}^k .

Equivalently, any nonsingular matrix U (whose columns are the generator vectors) describes a lattice

$$\mathcal{L} = \{Um; \text{all } m \in \mathcal{Z}^k\}.$$

$\mathcal{Z} =$ all integers

The lattice \mathcal{L} forms a codebook which together with its Euclidean Voronoi partition of a lattice forms a quantizer of \mathbb{R}^k called a *lattice quantizer*. This quantizer has an infinite number of levels.

Lattice points constitute a regularly spaced array of points.

Any shift, rotation, or scaling of a lattice is also a lattice.

Given a bounded subset A of \mathbb{R}^k , the intersection $\mathcal{L} \cap A$ forms a quantizer, also called a lattice quantizer, which has only a finite number of levels.

There is a wide literature on lattices. See, e.g., books by Conway and Sloane and by Coxeter and papers and the references therein.

Advantages of lattice quantizers

- Parameters: scale and support region.
- There are efficient algorithms for finding nearest neighbors, indexing, and lossless coding lattice quantizers, which makes them attractive in some applications such as audio coding.
- Good theoretical approximations for performance – special case of high rate theory, can make Gersho style approximations rigorous (quantizer cells have equal volume \Rightarrow Csiszar arguments validate Gersho's entropy approximations)
- Approximately optimal if used with entropy coding: constant quantizer point density functions over finite volume regions.

Note that if the lattice \mathcal{L} is scaled down to $a\mathcal{L}$, $M(aL_0) = M(L_0)$ does not change but $V(L_0)$ shrinks and $D(Q)$ decreases as $V(L_0)^{2/k}$

Suppose that the input probability density is concentrated on a bounded set A . Then since N is large and L_0 small, $V(A) \approx NV(L_0)$ so that

$$D(Q) \approx M(L_0)V(A)^{2/k}N^{-2/k},$$

a common approximation for analyzing lattice quantization. Thus the *best* lattice quantizer is the one that minimizes $M(L_0)$ over all lattices.

Denote by L_0 be the Voronoi cell of a lattice \mathcal{L} with Euclidean centroid at the origin, with volume $V(L_0)$ and normalized moment of inertia $M(L_0)$. Both are tabulated for the most important lattices and there exist algorithms for their evaluation.

In all of these examples of lattice quantizers, the high rate approximation simplifies because $M(S_i) = M(L_0)$ for all i and hence

$$\begin{aligned} D(Q) &\approx M(L_0) \sum_{i=1}^N f(\mathcal{D}(i))V(L_0)^{1+2/k} \\ &= M(L_0)V(L_0)^{2/k} \sum_{i=1}^N f(\mathcal{D}(i))V(L_0) \\ &\approx M(L_0)V(L_0)^{2/k} \end{aligned}$$

As an example, suppose the sequence of lattice quantizers is formed for a bounded set A by simply scaling down the generator matrix, e.g., for a given lattice \mathcal{L} form a sequence of codebooks $\frac{1}{n}\mathcal{L} \cap A$. Since a lattice has a regular array of points, the asymptotic fraction of points in A that lie in any subset S of A will be proportional to the ratio $V(S)/V(A)$, so

$$\Lambda(x) = \frac{1}{V(A)}; \quad x \in A.$$

High rate lattice quantizers have a uniform point density, but not vice versa. E.g., union of a bunch of lattices is not in general a lattice, but still scales down to uniform density.

High rate approximations for lattice quantization

Entropy:

$$H(q(X)) \approx h(X) - \log V(L_0)$$

Distortion:

$$\begin{aligned} D(q) &\approx M(L_0)V(L_0)^{2/k} \\ &\approx M(L_0)2^{2h(X)/k}2^{-2H(q(X))/k} \\ &= M(L_0)2^{2h(X)/k}2^{-R} \end{aligned}$$

- Compute $P_X(S_i)$ using model for density $f_X(x)$ & approximation

$$P_X(S_i) \approx f_X(y_i)V(L_0)$$

Use arithmetic code.

- Store optimal codeword lengths $l(\ell(i)) = -\log P_X(S_i)$ and centroids $\mathcal{D}(i)$ for a practical subset of is . Use formulas for other is .
- Need accurate model for f_X to be effective. E.g., Gaussian and Laplacian and generalized Gaussian popular.

Says that if N very large, then lattice codes chosen so that Voronoi cell has minimum normalized moment of inertia is nearly optimal in the ECVQ problem.

Implementation of lattice VQ

- Can compute nearest lattice point to any vector in \mathfrak{R}^k in constant time (independent of the number of lattice points)

Conway & Sloane (*IEEE Trans Inform Thy*, March 1982)

- Can compute index of nearest lattice point in constant time

Conway & Sloane (*IEEE Trans Inform Thy*, Nov 1983)

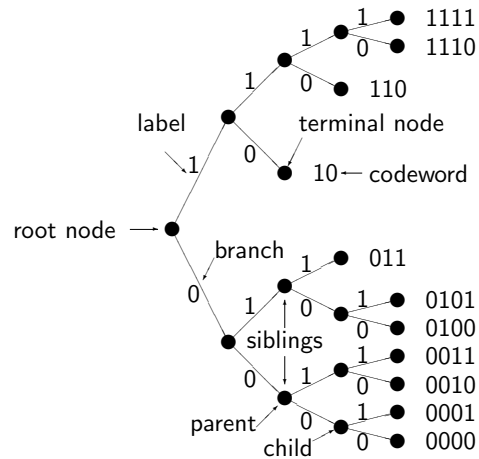
Nontrivial to index enormous codebook. Entropy coder converts indexes to actual channel codewords.

- Restrict to finite subset of lattice

Tradeoff: Lattice codes mean simple NN selection and nearly optimal variable rate performance, but need to entropy code to attain performance.

Tree-structured vector quantization

Constrain channel codewords to lie on branches of a tree



Can view each codeword as pathmap through the tree.

Can make code *progressive* and *embedded* by producing reproductions with each bit. Optimally done by centroid of node, conditional expectation of input given pathmap to that node.

Suggests suboptimal “greedy” encoder: Instead of finding best terminal node in tree (full-search), advance through the tree one node at a time, performing at each node a binary search involving only two children nodes.

Provides an approximate but fast search of the reproduction codebook: tree-structured vector quantization (TSVQ)

BFOS TSVQ design algorithm

Basic idea is taken from methods in statistics for designing decision trees: (See, e.g., *Classification and Regression Trees* by Breiman, Friedman, Olshen, & Stone)

- first *grow* a tree,
- then *prune* it.

trade off average distortion and average rate.

By first growing and then pruning back can get *optimal* subtrees of *good* trees.

Growing

Step 0 Begin with optimal 0 rate tree.

Step 1 “Split” node to form rate 1 bit per vector tree.

Step 2 Run Lloyd.

Step 3 If desired rate achieved, stop. Else either

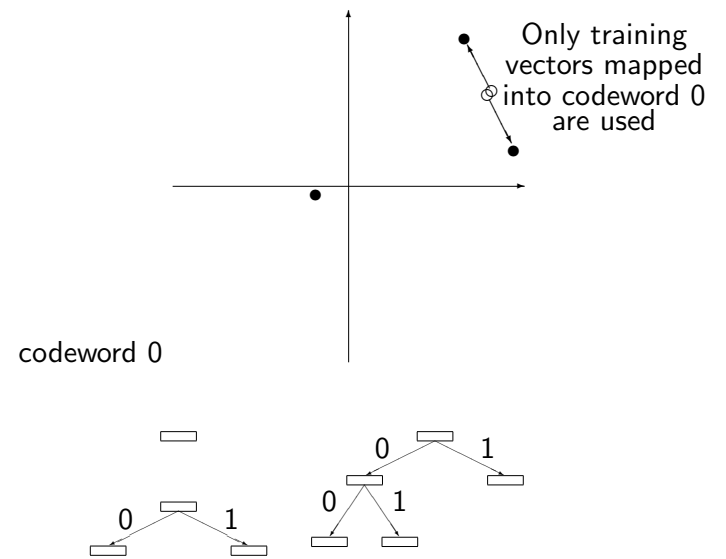
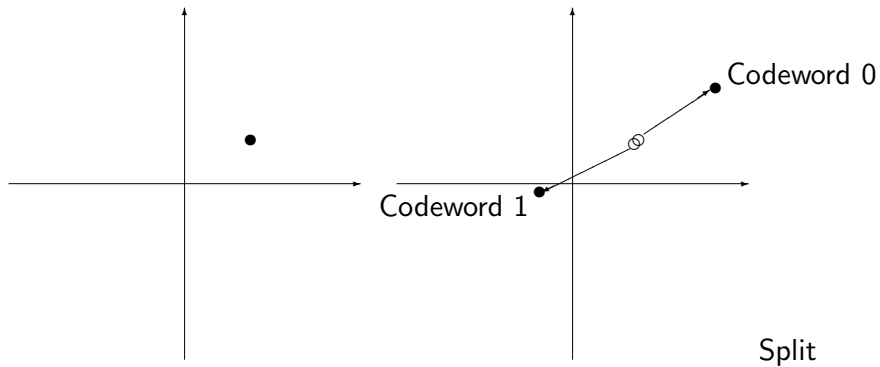
- Split all terminal nodes (balanced tree), or
- Split “worst” terminal node (largest conditional or partial distortion), or
- split in a greedy tradeoff fashion: maximize

$$\lambda(t) = \left| \frac{\text{change in distortion if split node } t}{\text{change in rate if split node } t} \right|.$$

Step 4 Run Lloyd.

Step 5 Go to Step 3.

Effect is that once a vector encoded into a node, it will only be tested against descendants of that node.



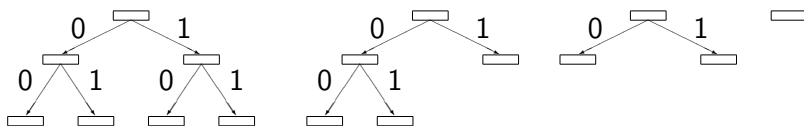
Once a tree has been *grown*, it can be *pruned*

Prune trees by finding subtree \mathcal{T} that *minimizes*

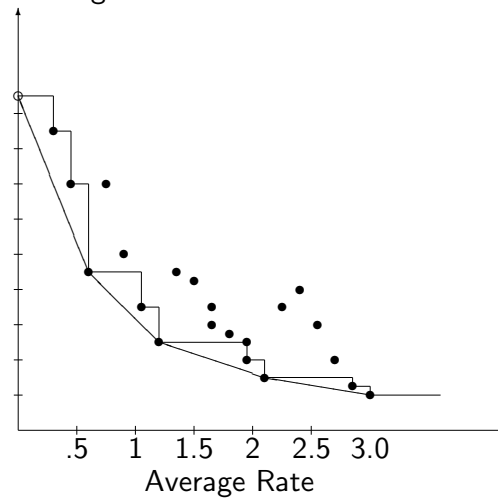
$$\lambda(\mathcal{T}) = \left| \frac{\text{change in distortion if prune subtree } \mathcal{T}}{\text{change in rate if prune subtree } \mathcal{T}} \right|.$$

- These optimal subtrees are nested

(BFOS Algorithm, extension of CART™ algorithm)

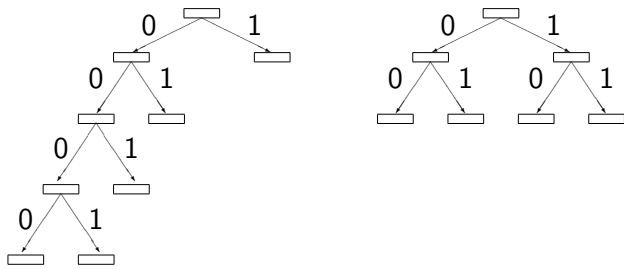


Average Distortion



BFOS finds codes on lower convex hull

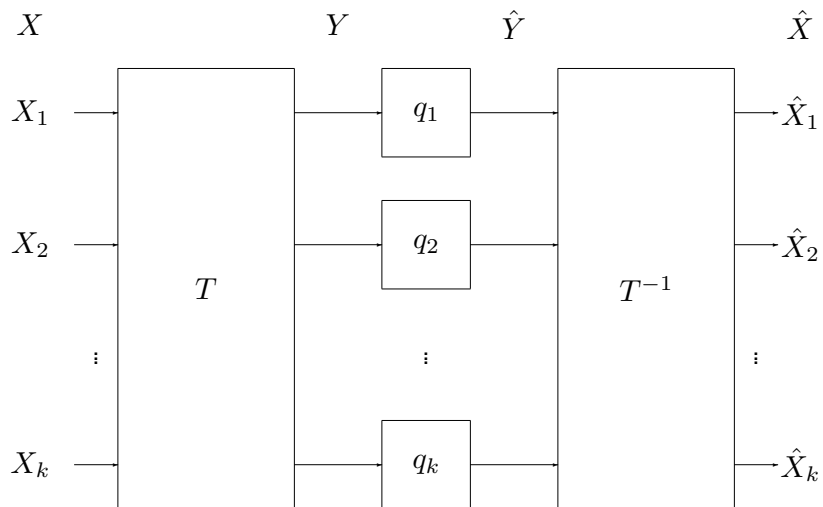
TSVQ Summary: balanced vs. unbalanced, fixed-rate vs. variable rate



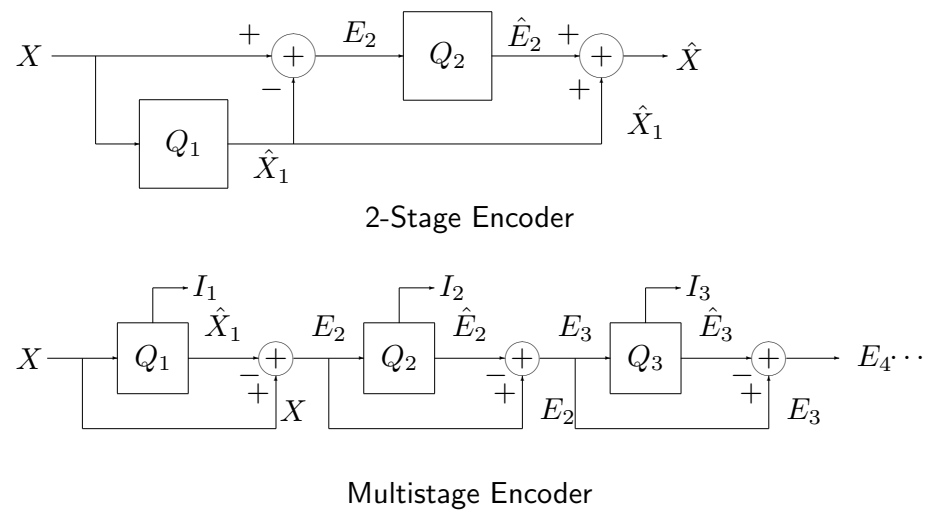
- Sequence of binary decisions. (Each node labeled, do pairwise minimum distortion.)
- Search is linear in bit rate (not exponential).
- Increased storage, possible performance loss

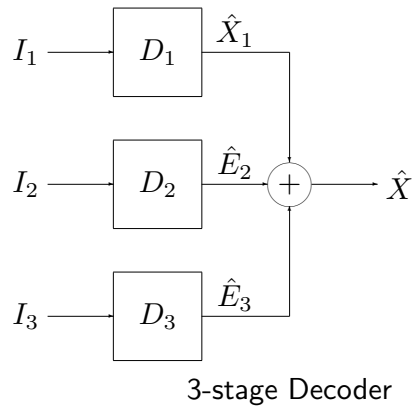
- Code is successive approximation (progressive, embedded)
- Table lookup decoder (simple, cheap, software)
- Fixed or variable rate

Transform Coding



Multistage VQ





Multistage a special case of TSVQ with a single codebook for each level.

See also residual VQ where allow smarter searches.

Product codes

Form a reproduction codebook for X as a Cartesian product of reproduction codebooks

E.g., application of a scalar codebook \mathcal{C}_m k times on a k -dimensional X , $\mathcal{C} = \times_{m=0}^{k-1} \mathcal{C}_m$

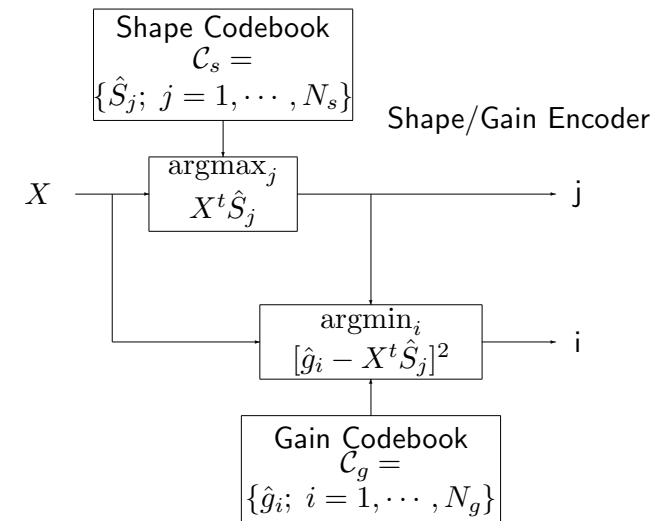
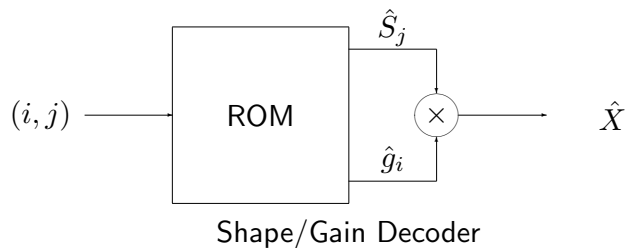
Lose optimality, but lower implementation complexity.

More interesting example:

Shape/Gain VQ (a Product Code)

Codebook: $\mathcal{C} = \mathcal{C}_g \times \mathcal{C}_s$

- $\mathcal{C}_g = \{\hat{g}_i; i = 1, \dots, N_g\}, \hat{g}_i \in (0, \infty)$
- $\mathcal{C}_s = \{\hat{S}_j; j = 1, \dots, N_s\}, \hat{S}_j \in \mathbb{R}^k, \|\hat{S}_j\|^2 = 1$



Note: Do not normalize input, real-time division a bad idea. No need, maximum correlation provides minimum MSE.

Two step encoding is optimal for given product code structure! Minimizes $d(X, \hat{g}_i \hat{S}_j)$ over all (i, j) :

$$\begin{aligned} \operatorname{argmin}_{i,j} \|X - \hat{g}_i \hat{S}_j\|^2 &= \operatorname{argmin}_{i,j} \left(\|X\|^2 - 2\hat{g}_i X^t \hat{S}_j + \hat{g}_i^2 \|\hat{S}_j\|^2 \right) \\ &= \operatorname{argmin}_{i,j} \left(-2\hat{g}_i X^t \hat{S}_j + \hat{g}_i^2 \right) \\ &= \operatorname{argmin}_i \left(-2\hat{g}_i \operatorname{argmax}_j X^t \hat{S}_j + \hat{g}_i^2 \right) \end{aligned}$$

First choose $j^* = \operatorname{argmax}_j X^t \hat{S}_j$, then choose $i^* = \operatorname{argmin}_i \left(-2\hat{g}_i X^t \hat{S}_{j^*} + \hat{g}_i^2 \right) = \operatorname{argmin}_i (\hat{g}_i - X^t \hat{S}_{j^*})^2$

Lloyd Codebook Design for Shape-Gain VQ

Training set $\{x_k\}$. Consider fixed-rate case.

Find the shape and gain codebooks that minimize the average distortion incurred in encoding the training vectors using codebook of given structure.

Fixed-rate shape-gain VQ described by three objects:

- Gain codebook $\mathcal{C}_g = \{\hat{g}_i; i = 1, 2, \dots, N_g\}$,
- Shape codebook $\mathcal{C}_s = \{\hat{S}_j; j = 1, 2, \dots, N_s\}$, and
- partition $\mathcal{P} = \{P_{i,j}; i = 1, 2, \dots, N_g; j = 1, 2, \dots, N_s\}$ of $N_g \times N_s$ cells describing the encoder:

Overall scheme is suboptimal because of constrained code structure, but better in practice because allows higher dimensional VQ for shape.

(Included in LPC and derivative speech coding standards.)

$x \in P_{i,j} \Rightarrow x$ is mapped into (i, j) , resulting reproduction is formed from the shape-gain vector (\hat{g}_i, \hat{S}_j) .

Express as $g(x) = \hat{g}_i$ and $S(x) = \hat{S}_j$.

Lloyd algorithm: Initialize these three components, then iteratively improve them by optimizing each in turn for the other two.

Define gain partition

$$\mathcal{G} = \{G_i; i = 1, \dots, N_g\}$$

by

$$G_i = \bigcup_{j=1}^{N_s} P_{i,j}.$$

G_i = region of the input vector space that maps into the gain codeword \hat{g}_i .

Define shape partition

$$\mathcal{A} = \{A_j; j = 1, \dots, N_s\}$$

by

$$A_j = \bigcup_{i=1}^{N_g} P_{i,j},$$

region mapping into the shape codeword S_j .

Note: Not true in general that $\mathcal{P} = \mathcal{G} \times \mathcal{A}$.
(Would be true if *independently* chose shape and gain.)

Denote average distortion by

$$D(\mathcal{C}_g, \mathcal{C}_s, \mathcal{P}) = E[d(X, g(X)S(X))].$$

Optimum gain codebook for fixed shape codebook and partition: Another necessary condition for optimality can be found by conditioning the input on the gain cells alone.

For any partition \mathcal{P}

$$\begin{aligned} D(\mathcal{C}_g, \mathcal{C}_s, \mathcal{P}) &= \sum_{i=1}^{N_g} E[d(X, \hat{g}_i S(X) | X \in G_i)] \Pr(X \in G_i) \\ &\geq \sum_{i=1}^{N_g} \inf_{g \geq 0} E[d(X, g S(X) | X \in G_i)] \Pr(X \in G_i). \end{aligned}$$

Value of g which yields the infimum for a particular i and \mathcal{P} (which determines the mapping $S(x)$) can be thought of as a *conditional gain centroid*, where the conditioning is on the partition \mathcal{P} and hence on the shape encoding.

Optimum partition for fixed gain and shape codebooks: For given \mathcal{C}_s and \mathcal{C}_g , let $\mathcal{P}^*(\mathcal{C}_s, \mathcal{C}_g)$ denote the minimum distortion partition (with the usual arbitrary tie-breaking rule).

For any encoder partition \mathcal{P}

$$D(\mathcal{C}_g, \mathcal{C}_s, \mathcal{P}) \geq D(\mathcal{C}_g, \mathcal{C}_s, \mathcal{P}^*(\mathcal{C}_s, \mathcal{C}_g)). \quad (80)$$

i.e., minimum distortion selection of gain and shape together minimizes the average distortion for a fixed pair of codebooks.

Assume for the moment that conditional centroids can be evaluated and denote the collection by

$$\mathcal{G}^*(\mathcal{C}_s, \mathcal{P}) = \{\hat{g}_i; i = 1, 2, \dots, N_g\},$$

– note the conditional gain centroids depend both on the shape codebook and on the partition \mathcal{P} .

Thus have the condition

$$D(\mathcal{C}_g, \mathcal{C}_s, \mathcal{P}) \geq D(\mathcal{G}^*(\mathcal{C}_s, \mathcal{P}), \mathcal{C}_s, \mathcal{P}). \quad (81)$$

Optimum shape codebook for fixed gain codebook and partition:

Similarly condition on the shape partition \mathcal{A} and write

$$\begin{aligned} D(\mathcal{C}_g, \mathcal{C}_s, \mathcal{P}) &= \sum_{j=1}^{N_s} E[d(X, g(X)\hat{S}_j)|X \in A_j] \Pr(X \in A_j) \\ &\geq \sum_{j=1}^{N_s} \inf_s E[d(X, g(X)s)|X \in A_j] \Pr(X \in A_j), \end{aligned}$$

– defines a conditional shape centroid given the overall partition.

Denote conditional shape centroids by

$$\mathcal{A}^*(\mathcal{C}_g, \mathcal{P}) = \{\hat{s}_j; j = 1, 2, \dots, N_s\},$$

Note that, as with the usual Euclidean centroid, must compute a vector sample average over a partition cell. Here, however, must normalize the average to have unit Euclidean distance.

Note: Normalization/division is *off line*, only in code design, not in code implementation!

Gain centroid is

$$\hat{g}_i = E(X^t S(X)|X \in G_i), \quad (84)$$

\Rightarrow

$$\hat{g}_i = \frac{1}{\|G_i\|} \sum_{j=1}^{N_s} \sum_{k: x_k \in P_{i,j}} \hat{s}_j^t x_k,$$

which is again the sample average. Obviously the gain centroids should be set to 0 if the above averages produce a negative number.

then have condition

$$D(\mathcal{C}_g, \mathcal{C}_s, \mathcal{P}) \geq D(\mathcal{C}_g, \mathcal{A}^*(\mathcal{C}_g, \mathcal{P}), \mathcal{P}). \quad (82)$$

Again note that the shape centroids depend on the gain codebook and the full partition.

All that remains is to compute the conditional centroids. Similar to Lloyd squared-error arguments shape centroid is given by

$$\hat{s}_j = \frac{E[g(X)X|X \in A_j]}{\|E[g(X)X|X \in A_j]\|}, \quad (83)$$

which becomes

$$\hat{s}_j = \frac{\sum_{i=1}^{N_g} \sum_{k: x_k \in P_{i,j}} g(x_k) x_k}{\|\sum_{i=1}^{N_g} \sum_{k: x_k \in P_{i,j}} g(x_k) x_k\|}.$$

Equations (81)–(82) and the centroids of (83)–(84) provide complete set of Lloyd conditions.

Shape-Gain VQ Design

Step 0. Initialization: Given $N_g, N_s, \epsilon > 0$, an initial product codebook $\mathcal{C}_g(0) \times \mathcal{C}_s(0)$, and a training set \mathcal{T} , set $m = 0$ and $D_{-1} = \infty$.

Step 1. Find the optimum partition $\mathcal{P}^*(\mathcal{C}_g(m), \mathcal{C}_s(m))$ of \mathcal{T} .

Step 2. Compute the average distortion

$$D_m = D(\mathcal{C}_g(m), \mathcal{C}_s(m), \mathcal{P}^*(\mathcal{C}_g(m), \mathcal{C}_s(m))).$$

If

$$(D_{m-1} - D_m)/D_m < \epsilon,$$

halt with $(\mathcal{C}_g(m), \mathcal{C}_s(m), \mathcal{P}^*(\mathcal{C}_g(m), \mathcal{C}_s(m)))$ describing the final quantizer. Else continue.

Step 3. Compute the optimum shape codebook using (83)

$$\mathcal{C}_s(m+1) = \{\hat{s}_j\} = A^*(\mathcal{C}_g(m), \mathcal{P}^*(\mathcal{C}_g(m), \mathcal{C}_s(m))).$$

Step 4. Compute the optimum partition $\mathcal{P}^*(\mathcal{C}_g(m), \mathcal{C}_s(m+1))$.

Step 5. Compute the optimum gain codebook using (84)

$$\mathcal{C}_g(m+1) = \{g_i\} = G^*(\mathcal{C}_s(m+1), \mathcal{P}^*(\mathcal{C}_g(m), \mathcal{C}_s(m+1))).$$

Step 6. Set $m = m + 1$ and go to Step 1.

Note: By handling gain separately, smaller codebooks and hence larger dimensions can be handled for shape. Result is this suboptimal structure can yield an implementable code with better distortion than an “optimal” code of comparable complexity.

There are variations on this theme, e.g., mean/shape, mean/gain/shape, and other product code structures.