

Chapter 12

Vector Quantization

- The original image decomposed into n -D image vectors
 - an $n = l \times m$ block of pixel values
 - a 3-D vector formed from the RGB color components of an individual pixel
 - DCT coefficients used as the vector components
- Each image vector X compared with a collection of codevectors, $\hat{X}_i, i = 1, \dots, N_c$, from a previously trained codebook and the best matching codevector \hat{X}_k is chosen

such that

$$d(X, \hat{X}_k) \leq d(X, \hat{X}_j), \quad j = 1, \dots, N_c$$

$$d(X, \hat{X}) = \frac{1}{n} \sum_{i=1}^n (x_i - \hat{x}_i)^2$$

$$d(X, \hat{X}) = \frac{1}{n} \sum_{i=1}^n w_i (x_i - \hat{x}_i)^2$$

- After a minimum distortion codevector has been selected, its index k is transmitted using $\log_2 N_c$ bits.
- At the receiver, this index is used as an entry to a duplicate codebook (a look-up table) to reproduce \hat{X}_k

- Compression: relatively few codevectors compared to $\#(\text{possible image vectors})$
 - Bit rate of VQ:

$$R = (\log_2 N_c)/n \quad \text{bits/pixel}$$

- For a given average distortion, D , it can achieve compression with the lowest bit rate
- For a given average bit rate, R , it can achieve compression with the lowest distortion

- In practice, the large values of n needed to approach the rate-distortion bound make codebook storage and searching impractical
 - Reasonable performance can be achieved with vectors of modest dimension
- The effect of vector dimension:
 - VQ encoding of 8-bit image at 1.0 bit/pixel
 - *

$$\left. \begin{array}{l} 2 \times 2 \text{ blocks } (n = 4) \\ N_c = 16 \end{array} \right\} R = \log_2 16/4 = 1.0 \text{ bit/pixel}$$
 - * 16 codevectors represent 256^4 possible image vectors of size 2×2

*

$$\left. \begin{array}{l} 4 \times 4 \text{ blocks } (n = 16) \\ N_c = 2^{16} \end{array} \right\} R = \frac{\log_2 2^{16}}{16} = 1.0 \text{ bit/pixel}$$

* 256^{16} possible image vectors of size 4×4

*

$$\frac{2^{16}}{2^{8 \times 16}} < \frac{2^4}{2^{8 \times 4}} \left(\frac{\# \text{ codevectors}}{\# \text{ possible image vectors}} \right)$$

*

$$\left\{ \begin{array}{l} \left(\frac{\# \text{ codevectors}}{\# \text{ possible image vectors}} \right) \downarrow \text{ the block size } \uparrow \\ MSE \downarrow \text{ the block size } \uparrow \end{array} \right.$$

- codebook generation
- codebook design for efficient searching and good performance

- Codebook Generation
 - Linde-Buzo-Gray (LBG) algorithm
 - * a training set of images representative of images to be encoded
 - * not requiring any information about the underlying image statistics
 - * local codebook:
 - optimal
 - computationally intensive
 - local codebook as overhead information (additional bits)
 - * global codebook:
 - several images as a training set
 - same class of imagery \Rightarrow good performance

- * as large and diverse a training set as possible to achieve reasonable average performance
- LBG algorithm is a generalization of the Lloyd-Max algorithm (1957, 1982) for scalar quantization:
 1. Given a set of training vectors, an initial codebook $\hat{X}_i^{(1)}$, $i = 1, \dots, N_c$, a distortion measure d , and a fractional distortion change ε ;
 $l \leftarrow 1$;
 $D^{(0)} \leftarrow$ a very large number

2. Map X_k to $\hat{X}_i^{(l)}$ if

$$d(X_k, \hat{X}_i^{(l)}) = \min_j d(X_k, \hat{X}_j^{(l)})$$

* compute the average distortion $D^{(l)}$

* if

$$\frac{D^{(l-1)} - D^{(l)}}{D^{(l-1)}} \leq \varepsilon,$$

the algorithm converges and thus terminates

3. Update the codebook:

$\hat{X}_i^{(l)}$ replaced by $\hat{X}_i^{(l+1)}$ minimizing the quantization error within the i -th decision region or cluster.

* If MSE is used, then the mean of the i -th decision region is $\hat{X}_i^{(l+1)}$;

$l \leftarrow l + 1$;

Go to Step 2.

Example

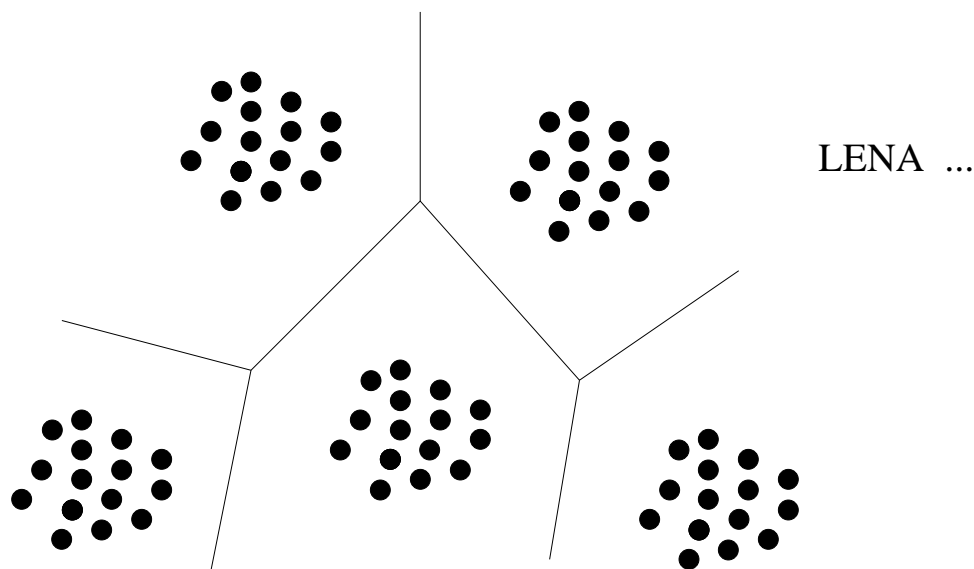
- LENA image $\rightarrow 1 \times 2$ blocks as training vectors
- a codebook of 8 codevectors
- Fig 12.2 (a), (b):
 - 1.50 bits/pixel
 - MSE = 72.53
- poor performance due to
 - * small vector dimension
 - * small codebook size

- Codebook initialization
 - LBG initial-dependent \Rightarrow a locally optimal codebook
 - Start with same simple codebook of correct size, or
 - Start with a simple smaller codebook and recursively construct larger ones till the desired size is reached
 - Rand Codes (Gray, 1984):
a random initial codebook
 - Splitting (Linde, Buzo and Gray, 1980):
The optimum code of one stage is split to form an initial code for the next stage

- Pairwise nearest neighbor (PNN) clustering (Equitz, 1989):
Merge two closest clusters of one stage to form an initial code for the next stage
- Codebook Design: Tree-Structured Codebooks
 - a full search of the codebook can be performed at a computational cost of $O(nN_c)$ with an associated storage cost nN_c

- To reduce the searching time, a tree structure can be imposed on the codebook, where each node has m branches and there are $p = \log_m N_c$ levels to the tree (Gray, 1984; Makhoul, 1985):
The computational cost reduces to $O(nm \log_m N_c)$ but the storage cost increases to $nm(N_c - 1)/(m - 1)$
- The codebook is designed by using successive applications of the LBG algorithm at each node for a codebook of size m

- A tree-structured codebook can never perform better than a single-level full search codebook
 - By increasing $\#$ (branches) at each node, performance can be improved and storage requirements can be reduced.
- Clustering: Gaussian Mixture Density Modeling and Decomposition
 - Clusters recognized as different classes
 - Piecewise Linear Classifier (Tree-Structured) based on labeled classes information



- Codebook Design: Product Codes
 - For a constant bit rate $R = (\log_2 N_c)/n$, the performance of VQ improves as the block size n increases. But, the codebook size $N_c = 2^{Rn}$ grows exponentially with n
 - A solution is to use a codebook with a product structure, i.e., a codebook that is formed as the Cartesian product of several smaller codebooks (Gray, 1984)
 - * A vector characterized by certain independent features, each feature encoded by a separate codebook
 - * The final codeword is the concatenation

- * For example, two separate codebooks of sizes N_1 and N_2 used to encode the orientation and the magnitude of a vector:
 - The effective size of the product codebook:
 $N_c = N_1 N_2$
 - The storage and computational complexity $\sim N_1 + N_2$
 - Certain image features have stable distributions, not overly sensitive to the particular input image

- Product codes can potentially outperform full-search codebooks with the same complexity and bit rate due to:
 - * The effective size of a product codebook is much larger than a full-search codebook
 - * The product codebook can be used to encode vectors of larger dimension while achieving the same bit rate
 - mean/residual VQ (M/R VQ)
 - interpolative/residual VQ(I/R VQ)
 - gain/shape VQ (G/S VQ)
 - classified VQ (C VQ)
 - finite-state VQ (FS VQ)

- Mean/Residual VQ
 - The original image divided into nonoverlapping blocks of size n (typically, $n = 4 \times 4 = 16$)
 - The mean computed for each block
 - Block means quantized using a scalar quantizer (typically, 8 bits) and transmitted to the receiver
 - DPCM can also be used to encode means
 - Residual vectors formed with approximately zero mean
 - Residual vectors quantized by VQ and the matching codevectors indices transmitted

- Reconstruction performed by adding means to decoded residual vectors
- Interpolative/Residual VQ
 - Reducing blocking artifacts by using a relatively smooth prediction image rather than the replicated block means
 - better reconstructed image quality than M/RVQ for the same bit rate
- The original image subsampled by l (typically, $l = 8$) in each dimension
- Each subsampled value quantized using a scalar quantizer (typically, 8 bits) and transmitted to the receiver

- A bilinear interpolative prediction of the original image generated at both the receiver and the transmitter using the quantized subsampled values and a residual image formed
- The residual image segmented into nonoverlapping blocks of size n (typically, $n = 4 \times 4 = 16$) forming vectors
- Residual vectors quantized using VQ and indices of best matching codevectors transmitted to the receiver
- Prediction image + decoded residual vectors: reconstruction

- Gain/Shape VQ

- The original image divided into nonoverlapping blocks of size n (typically, $n = 4 \times 4 = 16$) forming image vectors
- A unit energy shape codevector Y_j chosen to best match the image vector

$$(Y_j = \arg \max_{Y_k \in \text{the shape codebook}} X^T Y_k)$$

–

$$\sigma_i = \arg \min_{\sigma_k \in \text{the scalar gain codebook}} d(X, \sigma_k Y_j)$$

$$\hat{X} = \sigma_i Y_j : \text{the reproduction vector}$$

- (j, i) transmitted to the receiver
- Reconstruction: $\hat{X} = \sigma_i Y_j$

- Classified VQ (Ramamurthi and Gersho,1986)
 - The original image divided into nonoverlapping blocks of size n (typically, $n = 4 \times 4 = 16$)
 - Each block classified into one of M classes using an edge-oriented classifier:
 - * shade blocks (no significant gradient)
 - * midrange blocks (moderate gradient, but no definite edge)
 - * horizontal edge blocks

- * vertical edge blocks
 - * 45° edge blocks
 - * 135° edge blocks
 - * mixed blocks (edges with no discernible orientation)
- Each classified block encoded using the appropriate codebook
- * Codebooks of different sizes, $N_i, i = 1, \dots, M, N_c = \sum_i N_i$
 - * Each codebook may use a different distortion measure
 - * All possible codevectors labeled from 1 to N_c , the index of the chosen codevector transmitted

and used as an entry to a look-up table

* no overhead information required to indicate the class

- Finite-state VQ
 - VQ with memory
 - Finite-State machine (each state = a separate VQ codebook)
(Foster et al., 1985)
 - Correlation between adjacent blocks
- The original image divided into nonoverlapping blocks of size n (typically, $n = 4 \times 4 = 16$)
 - Blocks ordered, X_i , ($i = 0, 1, 2, \dots$)

- Given an initial state s_0 and its associated codebook C_{s_0} , encode X_0 by choosing an output codevector $\hat{X}_0 \in C_{s_0}$. Transmit the index of this codevector to the receiver
- Use the next-state function $f(\cdot)$ to determine the next state s_1 based upon s_0 and \hat{X}_0 , i.e., $s_1 = f(s_0, \hat{X}_0)$. Encode X_1 by $\hat{X}_1 \in C_{s_1}$ and transmit the index of \hat{X}_1
- Encode the remaining image vectors in the sequence by updating the state and using the associated codebook, i.e., $s_{n+1} = f(s_n, \hat{X}_n)$, $\hat{X}_{n+1} \in C_{s_{n+1}}$

- Since the next state is a function of the previous state and the output codevector, it is possible to track the state changes (or the quantizer choices) at the receiver without the use of overhead information
- VQ Results
M/R VQ and I/R VQ:
 - 4×4 blocks . a tree structure
 - each node: 8 branches(Oct-tree)
 - up to 5 levels for a final size of 2^{15} codevectors
 - incremental bit rate for each level: 3/16 bit/pixel
 - training set: 8 images of 512×512

LENA Third-Exam Project

- Implement Mean/Residual VQ
 - Codebook construction:
 - * Tree structure
 - * Five levels
 - * Each node has eight children
 - * LENA and ... \rightarrow nonoverlapping 4×4 blocks \rightarrow means and *residual vectors*
 - * Residual vectors used as a training set
 - * Each node associated with a subset of training set and further decomposed into eight subsets or clusters by using the LBG algorithm (training the tree)

- * For each node, the prototype of cluster is identified with a codeword
- * Only leaves, their prototypes are used at decoding (must be assigned indices used as entries to a lookup table at the decoding end)
- Scalar quantizer construction (for means):
 - DPCM on Mean Image of 128×128
 - Encoding each residual vector by the index of the leaf it settles
 - Decoding transmitted index as the entry to the corresponding prototype in the lookup table
 - Decoding DPCM
 - Reconstruction