Ch. 9  Vector Quantization (VQ)

Recall in lossless methods:
- Block Huffman is better than single-symbol Huffman; exploits correlation (assuming source symbols are not independent!)

Recall scalar quantization:
- it is the best lossy version of a single-symbol method

- Shannon proved that "blocking taken to the limit" achieves optimal compression; exploits correlation

This motivates Vector Quantization (VQ) not overlapping

- Group samples into vectors (blocks) and "quantize" the vector as a whole

Ex: 2-D Vectors:
Motivating Example: 2D Vector of Speech Signal

Equivalent to SQ

100 quantization cells

Only quantize this region with 100 smaller cells

Fewer than 100 cells of the same size

Improvement via VQ
Typically, vectors are formed from adjacent samples, but this is not necessary; let your *a priori* knowledge of the signal's correlation structure.

For Images - Vectors are typically formed using blocks of adjacent samples.

A: What does it mean to design a VQ?

A: Similar to SQ: Specify decision boundaries and reconstruction values in the N-D space; goal is minimize MSE for given rate.

Example of VQ Design: For 2-D vectors taken from sequence of independent Gaussian samples:
Each recon point (i.e. "quantized vector level") must be specified using a binary code word.

If \( M \) vectors are quantized using a VQ having \( M \) recon point we need code words having

\[
\text{Code word length} = \frac{L \log_2 M}{L} \text{ bits/vector}
\]

\[
\text{Bits/sample} = \frac{L \log_2 M}{L} \text{ bits/sample}
\]

"Typical" \( L \) values:
- Images: \( L = 16 \leq 4x4 \)
- Speech: \( L = 3, 4, 5, 6 \)
Info Theory Says: Increasing L Improves the VQ

Practice Says: ... to a point!

- Improvement decreases w/ ↑L
- Design harder w/ ↑L
- Encoder complexity grows w/ ↑L

Comparison of VQ Results with Published Results (Speech)

Solid Line: Published Results

o: Signal S1
x: Signal S2
*: Signal S3

2 bits/sample

VQ Dimension: L
Structure of a Vector Quantizer:
(Fig. 9.1 in textbook)
Encoder: Search Codebook for code vector \( \mathbf{y}_i \) that is closest to input vector \( \mathbf{x} \):

\[
\text{Codebook} = \{ \mathbf{y}_i \}_{i=1}^M = \mathbf{C}
\]

where each \( \mathbf{y}_i \in \mathbb{R}^L \)

\( \mathbf{x} \) is closest to \( \mathbf{y}_i \) if:

\[
\| \mathbf{x} - \mathbf{y}_i \|^2 \leq \| \mathbf{x} - \mathbf{y}_i \|^2 \quad \forall \mathbf{y}_i \in \mathbf{C}
\]

\[d(\mathbf{x}, \mathbf{y}_i) \leq d(\mathbf{x}, \mathbf{y}_j)\]

where \( \| \mathbf{z} \| = \sum_{i=1}^L z_i^2 \) (Euclidean Norm)

\[\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_L \end{bmatrix} \in \mathbb{R}^L\]

Notation: If \( \mathbf{x} \) is closest to \( \mathbf{y}_i \) we say

\[q(\mathbf{x}) = \mathbf{y}_i\]

Note: VQ cells are defined by the RL's rather than the boundaries.
Note: Encoder is computationally complex!

⇒ Must check all $M$ $Y_i$ for closeness
⇒ Compute $M$ norms

$M$ can be large (e.g. 256, 512, 1024)

Decoder: Use received binary codeword (index) as an address into the codebook (i.e. Table Look up)

⇒ Easy & Fast Computation!

Note: VA Complexity is asymmetrical

May not work well for real-time encoding