

Vector quantization

- **Scalar quantizers are special cases of vector quantizers (VQ): they are constrained to look at one sample at a time (memoryless)**
- **VQ does not have such constraint \Rightarrow better RD performance expected**
 - **Source coding theorems and other results in rate-distortion theory (due to Shannon) imply that one can always do better (in RD sense) if vector of samples are coded as units**
Note: We can code samples as units without necessarily exploiting or knowing interdependency between the samples
- **VQ widely used in coding speech, image, and video**

Vector quantization

- **Brief description**

- **Some main advantages:**

- ✓ exploit dependency that may exist within an input vector
 - ✓ ability to generate non-cubic multi-dimensional partitions of input which provides better compaction of the input space
 - ✓ ability to track high-order statistical characteristics of the input

- **Some main disadvantages:**

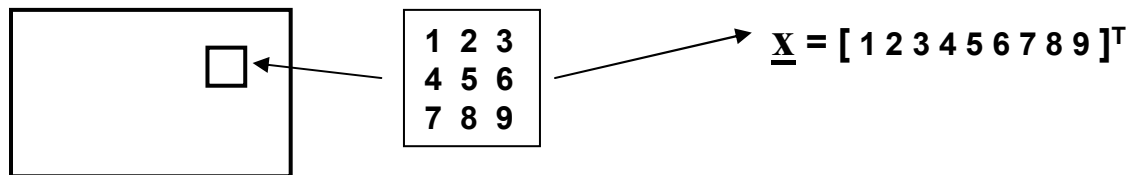
- ✓ encoding complexity and memory requirements increase exponentially with vector size (under a given rate) and with bit-rate
 - ✓ Lack of robustness: sensitivity to channel noise
 - ⇒ Conventional VQ is severely limited to modest vector and codebook size
 - ⇒ Different more robust methods needed

Vector quantization

- **Brief description**

- VQ takes blocks of pixels instead pixels

1. **Divide image into blocks (common size: 24x24, 16x16, 8x8)**



2. **Turn block into a vector**

3. **Compare \underline{x} with best matching $\hat{\underline{x}}$ in codebook**

Codebook: Table consisting of representative vectors (reconstruction levels) $\{\underline{r}_j\}_{j=1,\dots,M}$

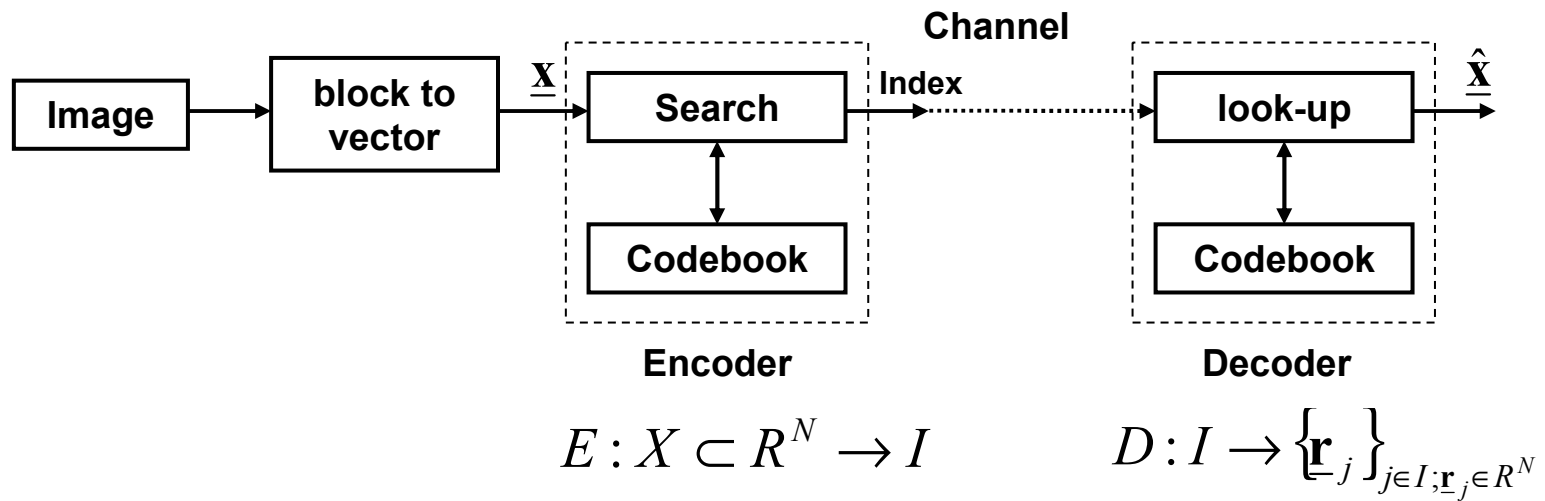
Best matching: with respect to a chosen distance (error) measure.

4. **Transmit the index “ k ” of that best matching vector: $\hat{\underline{x}} = Q(\underline{x}) = \underline{r}_k$**

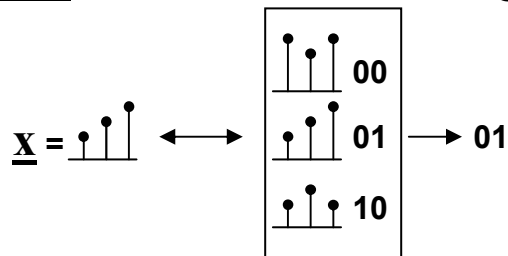
5. **Receiver gets the index “ k ” and retrieves $\hat{\underline{x}} = \underline{r}_k$ from its own stored codebook which matches transmitter’s codebook**

Vector quantization

- Brief description



Exp. Find best matching vector in codebook



Vector quantization

- **Motivation**

Theorem (from source coding and rate-distortion theory):

As vector size grows, performance improves in the rate-distortion sense

Practical constraints:

- **Encoding complexity and memory requirements increase exponentially with vector size (under a given bit-rate) and with bit-rate \Rightarrow codebook grows exponentially as a function of vector size N and bit-rate r .**
 - **Other problems: lack of robustness and sensitivity to channel noise**
- \Rightarrow Conventional VQ is severely limited to modest vector and codebook sizes**
- \Rightarrow Different more robust VQ approaches needed**

Vector quantization

- **VQ Design**

- In VQ, the N -dimensional input set $X \subset \mathbb{R}^N$ is divided into M regions or cells (“quantization levels”)

$$V_i = \{\underline{\mathbf{x}} \in X : Q(\underline{\mathbf{x}}) = \underline{\mathbf{r}}_i\}, \quad 1 \leq i \leq M$$

$\underline{\mathbf{r}}_i = i^{\text{th}}$ code vector (reconstruction level)

- **Optimal VQ**

- ✓ Let $d(\underline{\mathbf{x}}, \underline{\mathbf{y}})$ = defined distance measure between $\underline{\mathbf{x}}$ and $\underline{\mathbf{y}}$

Exp.: $\text{MSE} = E_{Q2} = E[d(\underline{\mathbf{x}}, \underline{\mathbf{y}})]; \quad d(\underline{\mathbf{x}}, \underline{\mathbf{y}}) = \sum_{i=1}^N (x_i - y_i)^2$

$$\text{MAE} = E_{Q1} = E[d(\underline{\mathbf{x}}, \underline{\mathbf{y}})]; \quad d(\underline{\mathbf{x}}, \underline{\mathbf{y}}) = \sum_{i=1}^N |x_i - y_i|$$

Vector quantization

Def: A vector quantizer is said to be optimal if the expected distortion

$$D = E[d(\underline{\mathbf{x}}, Q(\underline{\mathbf{x}}))] = \int d(\underline{\mathbf{x}}, Q(\underline{\mathbf{x}}))f(\underline{\mathbf{x}})d\underline{\mathbf{x}}$$

is minimized over all vector quantizers with M code vectors

➤ **Two necessary optimality conditions:**

1. **The nearest-neighbor condition:**

For a given set of code vectors $\{\underline{\mathbf{r}}_j\}_{j=1,\dots,M}$, $Q(\underline{\mathbf{x}})$ must be a nearest-neighbor mapping; i.e.:

$$Q(\underline{\mathbf{x}}) = \underline{\mathbf{r}}_i \text{ iff } d(\underline{\mathbf{x}}, \underline{\mathbf{r}}_i) \leq d(\underline{\mathbf{x}}, \underline{\mathbf{r}}_j); \text{ for } 1 \leq j \leq M$$

Vector quantization

2. The centroid condition:

For a given set of partition cells $\{V_j\}_{j=1,\dots,M}$, each code vector \underline{r}_j ($1 \leq j \leq M$) must be chosen so as to minimize the average distortion given a partition cell V_i ; i.e., \underline{r}_i is set to be the vector \underline{y} that minimizes the conditional distortion

$$D_i(\underline{y}) = E[d(\underline{x}, \underline{y}) | \underline{x} \in V_i] = \int_{\underline{x} \in V_i} d(\underline{x}, \underline{y}) f(\underline{x}) d\underline{x}$$

=> select \underline{r}_i ($1 \leq i \leq M$) such that

$$D_i(\underline{r}_i) = \min_{\underline{y}} D_i(\underline{y})$$

=> \underline{r}_i centroid of the cell V_i

Vector quantization

➤ Remarks

- ✓ Computation of a centroid for a particular cell depends on the distortion measure $d(\underline{x}, \underline{y})$.
- ✓ Another “less important” necessary condition for optimality is that for a given source distortion, points on the boundaries between nearest-neighbor cells occur with zero probabilities. This is automatically satisfied for continuous-valued input R.Vs.
- ✓ The “nearest-neighbor” and “centroid” conditions hold for scalar quantizers. They are very important because they are frequently used as the basis for most of the VQ design algorithms.

Vector quantization

➤ Remarks (continued)

- ✓ From above optimality conditions:

For given reconstruction levels (code vectors) $\{\underline{\mathbf{r}}_j\}_{j=1,\dots,M}$, the quantization levels are defined in terms of regions with “centroid” $\underline{\mathbf{r}}_j$ such that

$$V_i = \{ \underline{\mathbf{x}} : d(\underline{\mathbf{x}}, \underline{\mathbf{r}}_i) \leq d(\underline{\mathbf{x}}, \underline{\mathbf{r}}_j) \} ; \quad \forall j \in \{1, \dots, M\}$$

If MSE, $d(\underline{\mathbf{x}}, \underline{\mathbf{r}}_j) = |\underline{\mathbf{x}} - \underline{\mathbf{r}}_j|^2$ and

$$V_i = \{ \underline{\mathbf{x}} : |\underline{\mathbf{x}} - \underline{\mathbf{r}}_i|^2 \leq |\underline{\mathbf{x}} - \underline{\mathbf{r}}_j|^2 \}$$

Vector quantization

➤ **Remarks (continued)**

⇒ **Simple algorithm for performing VQ:**

1. For each input \underline{x} , compute distances

$$d(\underline{x}, \underline{r}_j) ; \forall j \in \{1, \dots, M\}$$

2. Choose “ i ” such that

$$d(\underline{x}, \underline{r}_i) \leq \min_{1 \leq j \leq M} d(\underline{x}, \underline{r}_j)$$

(Choose level corresponding to “closest” centroid)

Vector quantization

✓ Simple VQ Algorithm (continued)

- If more than one quantization level possible, use some predefined rule to make decision or simply make an arbitrary decision
- Computational requirements: if M code vectors (i.e., codebook has M entries), we have to compute M distances and make $M-1$ comparisons for each input sample \underline{x}
 - ⇒ This and the memory required to store the centroids put a limit on the practical size of the codebooks
- For given quantization levels $\{V_j\}_{j=1,\dots,M}$, the optimal reconstruction levels in the mean-square sense (i.e., $d(\underline{x}, \underline{r}_j) = |\underline{x} - \underline{r}_j|^2$) are

$$\underline{r}_j = \frac{\int_{V_j} \underline{x} f(\underline{x}) d\underline{x}}{\int_{V_j} f(\underline{x}) d\underline{x}}$$

Vector quantization

➤ Remarks (continued)

✓ Equations

$$V_i = \{ \underline{\mathbf{x}} : d(\underline{\mathbf{x}}, \underline{\mathbf{r}}_i) \leq d(\underline{\mathbf{x}}, \underline{\mathbf{r}}_j) \} \quad \text{and} \quad \underline{\mathbf{r}}_j = \frac{\int_{V_j} \underline{\mathbf{x}} f(\underline{\mathbf{x}}) d\underline{\mathbf{x}}}{\int_{V_j} f(\underline{\mathbf{x}}) d\underline{\mathbf{x}}}$$

can be used iteratively to design codebooks (find $\{\underline{\mathbf{r}}_j\}$) for vector quantizers that are optimal in the mse sense

⇒ most popular and classical VQ design technique is the Generalized Lloyd Algorithm (GLA), also known as the LBG (Linde, Buzo and Gray) algorithm (1981)

⇒ another widely used algorithm: Pairwise Nearest Neighbor (PNN) by Equitz (1989). Significantly reduces computation and no need for initial codebook, comparable reconstructed images quality.

Vector quantization

- **Generalized Lloyd (LBG) algorithm**
 - ✓ based on the optimality conditions mentioned earlier
 - ✓ most popular (although not best), widely used for comparison with other codebook design methods
 - ✓ adaptation of the “k-means” clustering algorithm

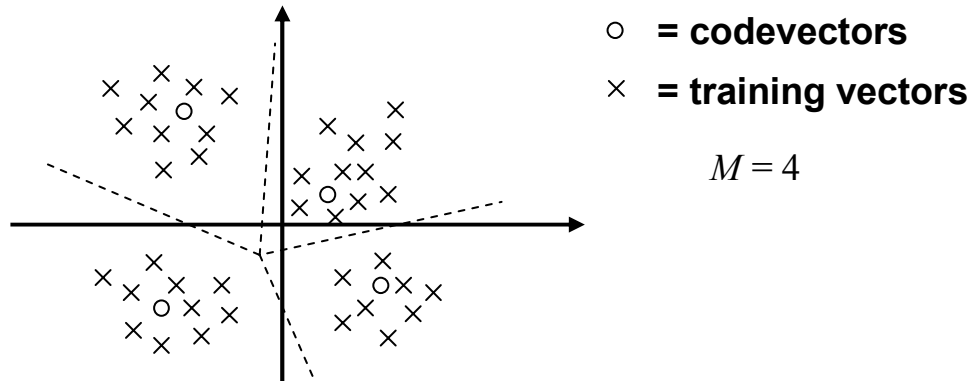
Basic steps:

Step 1: Start with a training set of vectors (get a large quantity of representative vectors: train on one set, test with others)

Step 2: Start with an initial codebook of size M (selected from training set); example: randomly selected vectors from training set

Step 3: Vector Quantize each training vector using current codebook (cluster training data)

Vector quantization



Step 4: Use centroid of clusters as the updated codebook

⇒ centroid = mean of cluster for mse and for a stationary and ergodic input since time/space averages replace statistical averages

⇒ centroid = center of mass

5. Repeat from Step 3 until distortion between old and new codebook is smaller than a selected small threshold

Vector quantization

Remarks on LBG

- ✓ At each iteration, the LBG algorithm constructs $\{\underline{\mathbf{r}}_j\}$ and $\{V_j\}$ satisfying

$$V_i = \{ \underline{\mathbf{x}} : d(\underline{\mathbf{x}}, \underline{\mathbf{r}}_i) \leq d(\underline{\mathbf{x}}, \underline{\mathbf{r}}_j) \} \quad \text{and} \quad \underline{\mathbf{r}}_j = \frac{\int_{V_j} \underline{\mathbf{x}} f(\underline{\mathbf{x}}) d\underline{\mathbf{x}}}{\int_{V_j} f(\underline{\mathbf{x}}) d\underline{\mathbf{x}}}$$

- ✓ LBG guaranteed to converge and finds a locally optimal quantizer for a training set (may not be locally optimal for the input $\underline{\mathbf{x}}$).
- ✓ Final resulting codebook depends on initial choice \Rightarrow algorithm influenced by choice of initial codebook (cluster centers), and by the choice and geometrical properties of training data.

Vector quantization

Remarks on LBG (continued)

- ✓ Local optimal design for fixed number of levels M . In coding, VQ usually used in conjunction with entropy coding
 - ⇒ limit the entropy of quantized signal rather than number of quantization levels in the design process
 - ⇒ entropy-constrained VQ (EC-VQ)

Vector quantization

Initialization in LBG

- ✓ **Most important issue since it can significantly affect the performance of designed codebook**
- ✓ **Several codebook initialization methods proposed.**
- ✓ **Popular ones:**
 1. **Random selection from training set**
 2. **Binary splitting for LBG codebook design**
 - » uses fixed perturbations of the current code vectors (centroids) to create more code vectors: twice as many at each step

Vector quantization

- **Basic steps of Binary Splitting for LBG:**
 1. **Step 1: Start with the centroid of the training set**
 2. **Step 2: Perturb the current centroid(s) (usually 2 opposite directions if size doubles at each iteration)**
 3. **Step 3: VQ all the training vectors, and take centroids of the new resulting clusters**
 4. **Step 4: Repeat Step 2 until we get the desired number of centroids (codevectors) for the initial codebook**
 5. **Do LBG**
- **Advantage: can reduce search complexity by using Tree search VQ instead of exhaustive search VQ**