Vector quantization

- **Example:** Binary Splitting, $M=8$

  - centroid of the entire training data
  - perturb centroid
  - cluster w.r.t. new centroids
  - compute new cluster centroids

  Advantage: Tree search can be used to reduce search complexity and computations
Vector quantization

- **Tree search VQ** (in general, does not have to be a binary tree)

  \[ M = 8 \]

  - 2 comparison vectors \[ 2 \text{ computations & 1 comparison} \]
  - 4 comparison vectors \[ 2 \text{ computations & 1 comparison} \]
  - 8 code vectors \[ 2 \text{ computations & 1 comparison} \]

  - In general:
    - \# computations = (\# splits at each node) \* (\# stages)
    - \# comparisons = (\# stages)

  - For binary tree: \# computations = \(2 \cdot \log_2(M) = 6\) for \(M = 8\)

  - Memory = \( \sum_{i=1}^{\text{#stages}} s^i = \frac{s - s^{(\text{#stages}+1)}}{1-s} \); where \(s = \# \text{ splits at each node}\)

  - Advantage: Reduces search complexity and computations
  - Disadvantages: Memory requirements \(\Rightarrow\) almost twice storage needed
Vector quantization

- **Exhaustive search VQ**
  (conventional unconstrained)

  \[ M = \text{codebook size} = 2^B \]
  \[ N = \text{Vector size} \Rightarrow N \text{ pixels/vector} \]

  \[ N \text{ pixels} \]

  \[ \begin{array}{c}
  1 \\
  2 \\
  \vdots \\
  M \\
  \end{array} \]

  \[ \Rightarrow \text{Memory} = M \times N = 2^B \times N \]

  where \( B \) = bits per codebook index (no entropy coding)

  \[ r = \text{bit rate} = \frac{\# \text{bits/pixel (sample)}}{\# \text{pixels per vector}} = \frac{B}{N} \]
Vector quantization

\[ r = \text{bit - rate} = \frac{\text{\# bits/pixel (sample)}}{\text{\# pixels per vector}} = \frac{\text{\# bits per codebook index}}{\text{\# pixels per vector}} = \frac{B}{N} = \frac{\log_2(M)}{N} \]

⇒ **Codebook size** \( M = 2^{rN} \) ⇒ exponential dependency on bit-rate \( r \) and vector size \( N \)

⇒ **Memory** = \( 2^{rN} \times N \)

\# distance computations = \( M = 2^{rN} \)

\# comparisons = \( M - 1 = 2^{rN} - 1 \)
Vector quantization

- Problem: As vector size $N$ increases, complexity increases much more than performance improves; performance increases at only algebraic rate, i.e. at a polynomial (linear, quadratic, …) rate
  - $\Rightarrow$ VQ limited to very small vectors ($N=4\times4=16$ is popular)
  - $\Rightarrow$ not much improvement in performance
  - $\Rightarrow$ complexity/performance tradeoffs are usually not good

- To overcome complexity barrier and eliminate exponential dependency, impose certain structural constrains on the VQ codebook
  - $\Rightarrow$ encoding complexities and/or memory requirements are algebraically dependent on bit-rate $r$ and/or vector size $N$
  - $\Rightarrow$ inferior RD performance for same $r$ and $N$ compared to unconstrained
  - $\Rightarrow$ reduction in complexity usually more than offsets the degradation in performance $\Rightarrow$ good complexity/performance tradeoffs
Vector quantization

Examples of Constrained VQs

- tree-structured VQs (TSVQ) (as known as multi-stage since search done in stages)
- Product VQs (e.g., Mean-extraction, Gain-shape)
- Lattice VQs
- Multistage Residual VQ (RVQ)

To improve performance without increasing vector size $N$, incorporate memory into VQ process.

Examples of constrained VQs with memory

- Finite-state VQ (FSVQ)
- Trellis VQ
Vector quantization

- Tree-structured VQ (TSVQ)
  - Codebook search and generation done in stages (multistage VQ)
  - Same as binary splitting but tree does not need to be binary
  - Employs a tree-structured VQ ⇒ search complexity becomes linear instead of exponential (but more memory needed)
  - Other practical advantages:
    ✓ Suitability for progressive transmission
    ✓ Lower sensitivity to channel noise
    ✓ Could be easily used as a component of a variable rate compression system
    ✓ Main weakness: memory requirements can be twice that of unconstrained VQ ⇒ still put severe limitation on vector size and/or bit-rates
Vector quantization

- **Mean-extraction VQ (Mean-residual VQ)**
  - Basic principle:
    - Remove the mean (or average, DC component)
    - Quantize and send the mean separately
    - Quantize the residual using a VQ codebook
      ⇒ helps to reduce codebook size
  - To encode:
    1. Subtract mean
    2. VQ
    3. Send mean (scalar quantized)
Vector quantization

• Gain-shape VQ
  ➢ Basic principle:
    ✓ like above, but normalize vectors and send a gain instead of mean. So here the gain (or “energy”) of the vector is computed and quantized
    ✓ Note: Can combine mean-extraction and gain-shape
Vector quantization

- **Multistage Residual VQ (RVQ)**
  - Commonly referred to as Residual VQ or Multistage VQ
  - Another less common name is Cascade VQ
  - Consists of several VQ stages with codebook designed to code residual vectors
  - Typical RVQ structure

- To encode: send addresses from each stage
- To reconstruct: add up the codevectors retrieved from corresponding address/codebooks
Vector quantization

- Overall codebook constrained to being the direct sum of smaller codebooks
- What's nice about RVQ?
  - Assume \( L \) stages
  - Assume each VQ stage has \( m \) codevectors
  - Overall codebook size:
    - By taking different combinations of codevectors from the different stages, one can represent \( m \times m \times \ldots \times m = m^L \) codevectors (may be large)
  - Required storage:
    - How many vectors to store? \( L \times m \) (may be manageable)

Result:
- We can represent \( m^L \) codevectors by storing only \( L \times m \) codevectors
- Ability of RVQ to employ larger codebooks
Vector quantization

- Finite-State VQ (FSVQ)
  - FSVQ is a finite-state machine with a VQ codebook for each state (multi-codebook approach)
  - FSVQ consists of a finite collection of VQs, where each successive source vector is encoded using a VQ codebook determined by the current encoder state

**Example**
Vector quantization

- For an input $x$, the index transmitted is the one that minimizes the distortion of the current state VQ codebook (nearest neighbor using current state VQ)
- Good quality can be achieved by exploiting correlation between adjacent blocks

**But:**
- Increase in memory required to store the VQ codebooks for all the states
- Generated state sequence (which minimizes current state VQ codebook distortion) does not necessarily result in minimization of the overall distortion introduced by the state sequence
Vector quantization

- Trellis VQ (corresponds to an FSVQ with a finite delay)
  - One way to overcome aforementioned problem is to use Trellis VQ, which allows “delayed decision encoding”, i.e., the FSVQ tries every possible sequence of finite length \( L \) and picks the one that results in the overall minimum distortion
  - A finite delay introduced to increase performance
  - Issues: Encoder complexity can become large
  - Solution: employ a small number of states in conjunction with predictive techniques (predictive trellis encoder).