Vector Quantization

Vectors
- An a x b block can be considered to be a vector of dimension ab.
  \[ \text{block} = (w,x,y,z) \text{ vector} \]
- Nearest means in terms of Euclidian distance or Euclidian squared distance. Both equivalent.
  \[ \text{Distance} = \sqrt{(w_w-w_j)^2 + (x_x-x_j)^2 + (y_y-y_j)^2 + (z_z-z_j)^2} \]
  \[ \text{Squared Distance} = (w_w-w_j)^2 + (x_x-x_j)^2 + (y_y-y_j)^2 + (z_z-z_j)^2 \]
- Squared distance is easier to calculate.

Vector Quantization Facts
- The image is partitioned into a x b blocks.
- The codebook has n representative a x b blocks called codewords, each with an index.
- Compression with fixed length codes is
  \[ \log_2 \frac{ab}{n} \text{ bpp} \]
- Example: \( a = b = 4 \) and \( n = 1,024 \)
  - Compression is \( 10/16 = .63 \text{ bpp} \)
  - Compression ratio is \( 8 : .63 = 12.8 : 1 \)
- Better compression with entropy coding of indices

Examples
- 4 x 4 blocks: .63 bpp
- 4 x 8 blocks: .31 bpp
- 8 x 8 blocks: .16 bpp
- Codebook size = 1,024

Scalar vs. Vector
- Pixels within a block are correlated.
  - This tends to minimize the number of codewords needed to represent the vectors well.
- More flexibility.
  - Different size blocks
  - Different size codebooks
**Encoding and Decoding**

- **Encoding:**
  - Scan the a x b blocks of the image. For each block find the nearest codeword in the codebook and output its index.
  - Nearest neighbor search.

- **Decoding:**
  - For each index output the codeword with that index into the destination image.
  - Table lookup.

**The Codebook**

- Both encoder and decoder must have the same codebook.
- The codebook must be useful for many images and be stored somehow.
- The codebook must be designed properly to be effective.
- Design requires a representative training set.
- These are major drawbacks to VQ.

**Codebook Design Problem**

- Input: A training set X of vectors of dimension d and a number n. (d = a x b and n is number of codewords)
- Output: n codewords c(0), c(1),...,c(n-1) that minimizes the distortion.
  \[ D = \sum_{x \in X} \| x - c(\text{index}(x)) \|^2 \]  
  sum of squared distances

where index(x) is the index of the nearest codeword to x.

\[ \| [x_0, x_1, \ldots, x_d] \|^2 = x_0^2 + x_1^2 + \cdots + x_{d-1}^2 \]  
 squared norm

**GLA**

- The Generalized Lloyd Algorithm (GLA) extends the Lloyd algorithm for scalars.
  - Also called LBG after inventors Linde, Buzo, Gray (1980)
  - It can be very slow for large training sets.

**GLA Example (1)**

GLA Algorithm:

Choose a training set X and small error tolerance \( \epsilon > 0 \).
Choose start codewords \( c(0), c(1), \ldots, c(n-1) \).
Compute \( X(j) := \{ x : x \text{ is a vector in } X \text{ closest to } c(j) \} \).
Compute distortion \( D \) for \( c(0), c(1), \ldots, c(n-1) \).
Repeat
  - Compute new codewords
    \[ c'(j) := \text{round} \left( \frac{1}{|X(j)|} \sum_{x \in X(j)} x \right) \text{ (centroid)} \]
  - Compute \( X'(j) := \{ x : x \text{ is a vector in } X \text{ closest to } c'(j) \} \).
  - Compute distortion \( D' \) for \( c'(0), c'(1), \ldots, c'(n-1) \).
  - If \( |(D - D')/D| < \epsilon \) then quit
  - else \( c := c'; X := X'; D := D' \).
End(repeat)
GLA Example (8)  
C S E  4 9 0  g z  - L e c t u r e  1 2  - W  i n t e r  2 0 0 4

GLA Example (9)  
C S E  4 9 0  g z  - L e c t u r e  1 2  - W  i n t e r  2 0 0 4

GLA Example (10)  
C S E  4 9 0  g z  - L e c t u r e  1 2  - W  i n t e r  2 0 0 4

C o d e w o r d  S p littin g
• It  is  p o s s ib l e  th a t a  c h o s e n  c o d e w o r d  r e p r e s e n t s  n o  tr a in in g  v e c to r s , th a t  is ,  X ( j)  i s  e m p t y .
  –  S p litti ng  i s  a n  a lte r n a t iv e  c o d e b o o k  d e s i g n  a l g o r i t h m  th a t  a v o i d s  th i s  p r o b l e m .
• B a s ic  I d e a
  – S e le c t  c o d e w o r d  c ( j)  w i t h  t h e  g r e a t e s t  d i s t o r t i o n .
  –  S p l i t  i t  i n t o  t w o  c o d e w o r d s  t h e n  d o  t h e  G L A .

C o d e w o r d
• I n i t i a l l y  c ( 0 )  i s  c e n t r o i d  o f  tr a in i n g  s e t

C o d e w o r d
• 1  x  2  c o d e w o r d s
  N o t e :  c o d e w o r d s  d i a g o n a l l y  s p r e a d

C o d e b o o k

E x a m p l e  o f  S p littin g

Example of Splitting
Example of Splitting

- Codeword
- Training vector
- Split $c(1) = c(0) + \epsilon$

**Example of Splitting**

- Codeword
- Training vector
- Apply GLA

**Example of Splitting**

- Codeword
- Training vector
- $c(0)$ has max distortion so split it.

**Example of Splitting**

- Codeword
- Training vector
- $c(2)$ has max distortion so split it.

**Example of Splitting**

- Codeword
- Training vector
- $c(3)$
GLA Advice

- Time per iteration is dominated by the partitioning step, which is \( m \) nearest neighbor searches where \( m \) is the training set size.
  - Average time per iteration \( O(m \log n) \) assuming \( d \) is small.
- Training set size.
  - Training set should be at least 20 training vectors per code word to get reasonable performance.
  - Too small a training set results in “over training”.
- Number of iterations can be large.

Encoding

- Naive method.
  - For each input block, search the entire codebook to find the closest codeword.
  - Time \( O(T n) \) where \( n \) is the size of the codebook and \( T \) is the number of blocks in the image.
  - Example: \( n = 1024 \), \( T = 256 \times 256 = 65,536 \) (2 x 2 blocks for a 512 x 512 image)
    \[ nT = 1024 \times 65536 = 2^{26} = 67 \text{ million distance calculations}. \]
- Faster methods are known for doing “Full Search VQ”. For example, k-d trees.
  - Time \( O(T \log n) \)