Institut
Mines-Telecom

Vector Quantization

Marco Cagnazzo,
cagnazzo@telecom-paristech.fr

MN910 – Advanced Compression

Outline

Outline

Scalar quantization (SQ)

Definition

$$Q : x \in \mathbb{R} \rightarrow y \in \mathcal{C} = \{\widehat{x}^1, \widehat{x}^2, \ldots \widehat{x}^L\} \subset \mathbb{R}$$

- Resolution (or rate) : $b = \log_2 L$ bits per sample
- Distortion (ideal case)

$$\sigma_Q^2 = c_X \sigma_X^2 2^{-2b}$$

- c_X (shape factor) is a constant i depending on X probability distribution
  - $c_X = 1$ for uniform variables
  - $c_X = \frac{\sqrt{3}}{2}\pi$ for Gaussian variables

Predictive quantization (PQ)

Performances

- In PQ, we quantize the difference Y between X and the linear prediction based on P past samples
- Prediction gain:

$$G_{\mathrm{P}}(P) = \frac{\sigma_X^2}{\sigma_Y^2} = \frac{\sigma_X^2}{[\det \boldsymbol{R}_P]^{1/P}}$$

- Asymptotic value of G_{P}

$$G_{\mathrm{P}} = \lim_{P \to +\infty} \frac{\sigma_X^2}{[\det \boldsymbol{R}_P]^{1/P}}$$

where $\boldsymbol{R}_P$ is the covariance matrix of $[X_1, X_2, \ldots, X_P]$

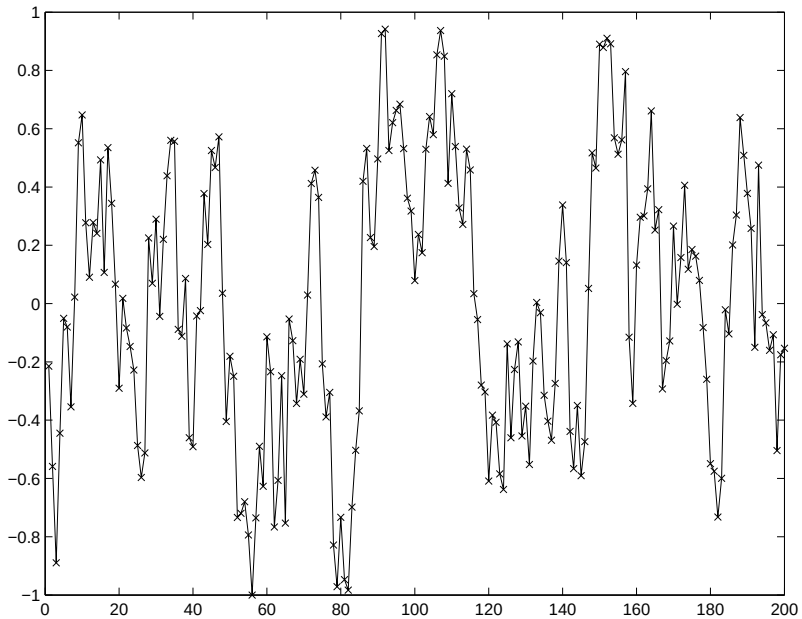
Vector quantization (VQ)

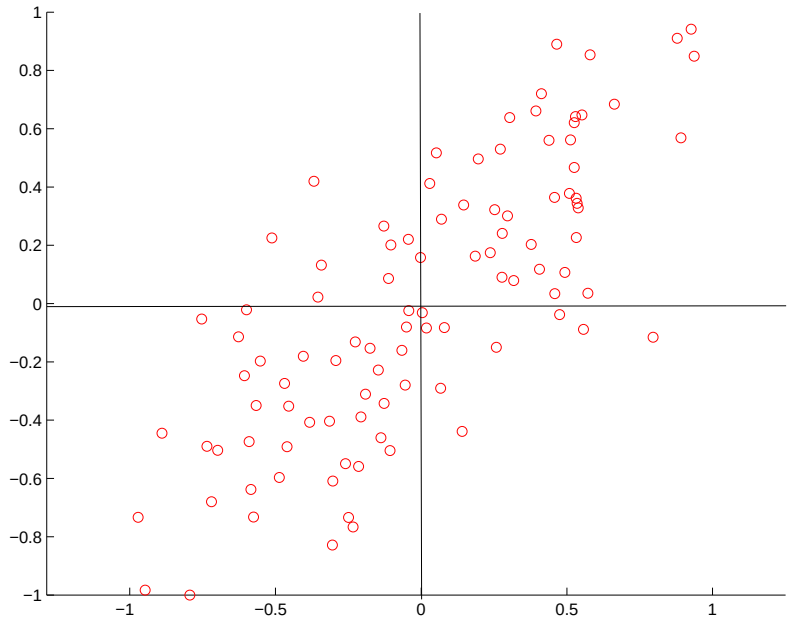
Introduction

- We typically consider *sequences* of samples, rather than an isolated sample
- Idea: to jointly quantify several samples, i.e. a *vector*
- The input space $\mathbb{R}^N$ is therefore partitioned into cells (decision regions)
- For each region we have a representative vector (codewords)

Example: correlated signal

AR(2) Signal





Vector quantization

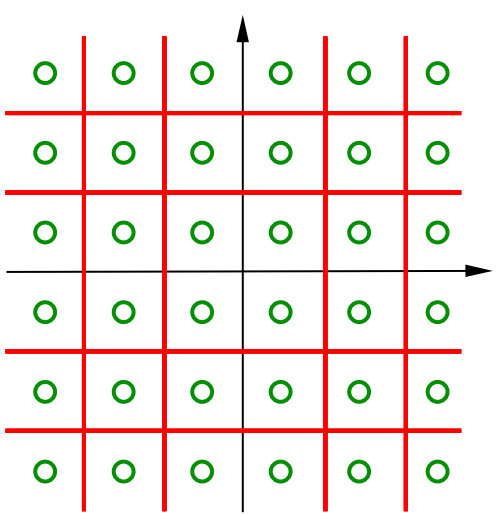
Introduction

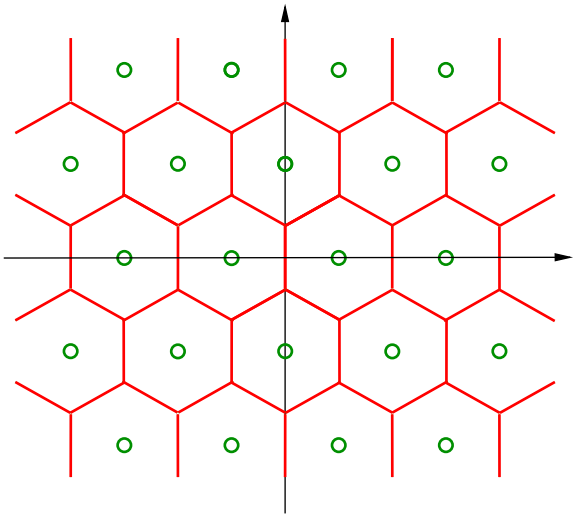
- Generalization of SQ to the space $\mathbb{R}^N$
- SQ can be seen as a special case of VQ
  - Cells are delimited by hyperplanes orthogonal to axes
  - Codewords are aligned to axes
- VQ potential gains
  - Geometric gain: arbitrary shape of cells and arbitrary position of codewords
  - Correlation gain: we exploit directly signal correlation

Properties of VQ

VQ improves with respect to SQ:

- we are able to exploit sample correlation
- we suppress the constraint of hypercubic cells
- we suppress the constraint of integer rate





Terminology

Definition of vector quantization (VQ):

$$Q : \underline{x} \in \mathbb{R}^N \rightarrow \underline{y} \in \mathcal{C} = \{\underline{\widehat{x}}^1, \underline{\widehat{x}}^2, \ldots \underline{\widehat{x}}^L\} \subset \mathbb{R}^N$$

- Space dimension: $N \in \mathbb{N}$
- Codebook size: $L \in \mathbb{N}$
- Resolution (or rate): $b = \frac{\log_2 L}{N} = \frac{R}{N}$

Terminology

$$Q : \underline{x} \in \mathbb{R}^N \rightarrow \underline{y} \in \mathcal{C} = \{\underline{\widehat{x}}^1, \underline{\widehat{x}}^2, \ldots \underline{\widehat{x}}^L\} \subset \mathbb{R}^N$$

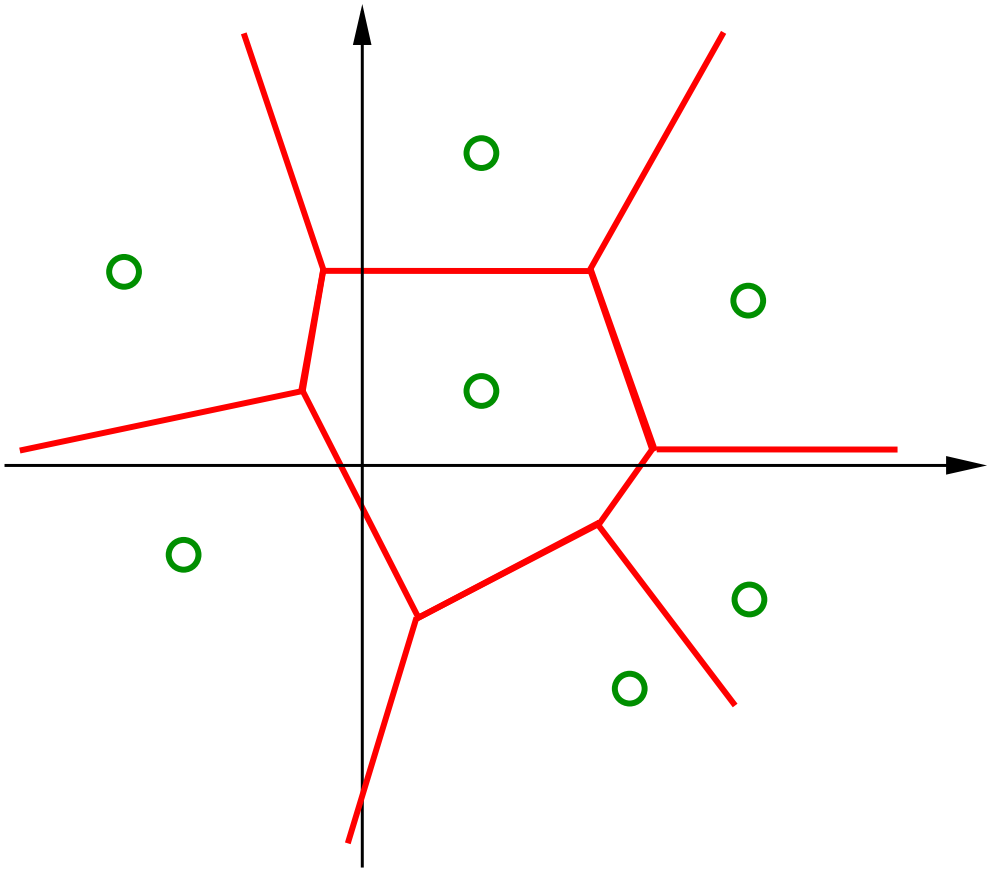
- Decision regions (or cells)

$$\Theta^i = \{\underline{x} : Q(\underline{x}) = \underline{\widehat{x}}^i\} \qquad \Theta^i \cap \Theta^j = \emptyset \qquad \cup_i \Theta^i = \mathbb{R}^N$$

- $\mathcal{C}$ is called codebook
- The elements of $\mathcal{C}$ are called *codewords*, *output vectors*, *reproduction vectors*
- Only *regular* quantifiers are considered. Regular VQ is characterized by:
  - Convexity : $\underline{x}_1, \underline{x}_2 \in \Theta^i \Rightarrow [\lambda \underline{x}_1 + (1-\lambda)\underline{x}_2 \in \Theta^i] \; \forall \lambda \in [0,1]$
  - Codeword within the region: $\forall i \in \{1, \ldots, L\}, \underline{\widehat{x}}^i \in \Theta^i$

Regular quantizer

- Convex regions
- $\underline{\hat{x}}^i \in \Theta^i$

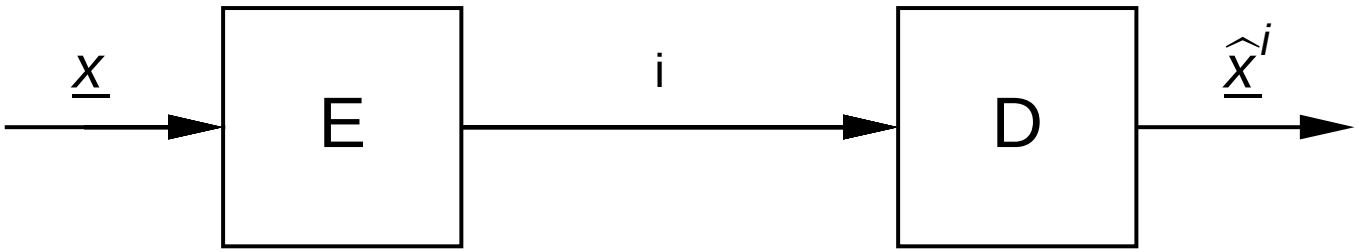


Vector quantization

VQ as encoding and decoding

Just as in the case of scalar quantization, we can interpret VQ as the cascade of two operations:

- *Encoding*: Input vector $\underline{x}$ is associated to index i (to cell Θ^i)
- *Decoding*: Index i (cell Θ^i) is associated to output vector (codeword) $\underline{\hat{x}}^i$



$\{\Theta^i\}$ $\mathcal{C} = \{\underline{\hat{x}}^i\}_i$

Quantization error

- Definition from scalar quantization (SQ) is generalized
- We use the joint probability density function (PDF) and the Euclidean norm:

$$\sigma_{\mathrm{Q}}^2 = \frac{1}{N} \int_{\mathbb{R}^N} \|\underline{x} - Q(\underline{x})\|^2 p_{\underline{X}}(\underline{x}) d\underline{x}$$

$$= \frac{1}{N} \sum_{i=1}^{L} \int_{\Theta^i} \|\underline{x} - Q(\underline{x})\|^2 p_{\underline{X}}(\underline{x}) d\underline{x}$$

$$= \frac{1}{N} \sum_{i=1}^{L} \int_{\Theta^i} \|\underline{x} - \underline{\widehat{x}}^i\|^2 p_{\underline{X}}(\underline{x}) d\underline{x}$$

Outline

Building the codebook

Problrme : Find the codebook $\mathcal{C}$ minimizing l'erreur σ^2_{Q}
Solution : Generalized Lloyd Algorithm (GLA)

1. Initialization : $k = 0$, we start with some codebook $\mathcal{C}_{(k)} = \{\underline{\widehat{x}}^0_{(k)}, \underline{\widehat{x}}^1_{(k)}, \ldots, \underline{\widehat{x}}^L_{(k)}\}$, we set $\sigma^2_{\mathrm{Q},(k-1)} = \infty$, we choose ϵ
2. We optimize cells with respect to codebook obtaining $\{\Theta^i_{(k)}\}_i$
3. We optimize the codebook with respect to cells, obtaining $\mathcal{C}_{(k+1)}$
4. We compute distortion $\sigma^2_{\mathrm{Q},(k)}$ associated to $\mathcal{C}_{(k+1)}$, $\{\Theta^i_{(k)}\}_i$
5. If $\frac{\sigma^2_{\mathrm{Q},(k-1)} - \sigma^2_{\mathrm{Q},(k)}}{\sigma^2_{\mathrm{Q},(k-1)}} < \epsilon$, we stop; else $k \leftarrow k + 1$, and go to step 2

Generalized Lloyd Algorithm (GLA)

- Cells optimization with respect to the codebook:

$$\Theta^i_{(k)} = \left\{ \underline{x} : \|\underline{x} - \underline{\widehat{x}}^i_{(k)}\| \leq \|\underline{x} - \underline{\widehat{x}}^j_{(k)}\| \ \forall j \in \{1, \ldots, L\} \right\}$$

- Nearest neighbor rule
- Complexity proportional to L

Generalized Lloyd Algorithm (GLA)

- Codebook optimization with respect to the cells: centroid rule
- For the sake of simplicity, $D = \sigma^2_{\mathrm{Q},(k)}$

$$D = \frac{1}{N}\sum_{i=1}^{L}\int_{\Theta^i} \|\underline{x} - \widehat{\underline{x}}^i\|^2 p_{\underline{X}}(\underline{x})d\underline{x} = \frac{1}{N}\sum_{i=1}^{L} D_i$$

$$\frac{\partial D_i}{\partial \widehat{x}^{i,j}} = \int_{\Theta^i} 2(x^j - \widehat{x}^{i,j}) p_{\underline{X}}(\underline{x})d\underline{x}$$

$$\widehat{x}^{i,j} = \frac{\int_{\Theta^i} x^j p_{\underline{X}}(\underline{x})d\underline{x}}{\int_{\Theta^i} p_{\underline{X}}(\underline{x})d\underline{x}}$$

$$\widehat{\underline{x}}^i = \mathrm{E}[\underline{x}|\underline{x} \in \Theta^i]$$

Generalized Lloyd Algorithm (GLA)

Bilan

- Necessity of estimate probability distributions
- Very high complexity - suboptimal algorithms
- Convergency to local minimum and impact of inizialization
- Simulated annealing algorithms (extremely high complexity, global optimum attained subject to some hypotheses with a given probability)

GLA initialization

Linde-Buzo-Gray (LBG) algorithm

- Iterative algorithm
- For $L = 1$, we apply GLA
- The resultat $\underline{\widehat{x}}_0^0$ is the centroid of population
- **Split** of $\mathcal{C}_0 = \{\underline{\widehat{x}}_0^0\}$: $\underline{\widehat{x}}_1^0 = \underline{\widehat{x}}_1^1 \quad \underline{\widehat{x}}_0^0 + \underline{\epsilon}$
- GLA on the new codebook $\mathcal{C}_1 = \{\underline{\widehat{x}}_1^0, \underline{\widehat{x}}_1^1\}$
- **Split** of $\mathcal{C}_1$: each codeword $\underline{\widehat{x}}_1^i$ gives $\underline{\widehat{x}}_2^{2i} = \underline{\widehat{x}}_1^i$ and $\underline{\widehat{x}}_2^{2i+1} = \underline{\widehat{x}}_1^i + \underline{\epsilon}_i$
- Iteration until we obtain L vectors

Data-based codebook

- In practice, we do not always have the PDFs
- We can build a *training set* (TS)
  - Sound samples, blocks of pixels from images, . . .
- The training set is used to create the codebook:
  - *Nearest neighbor*: TS vectors are associated to the cell represented by the nearest neighbor
  - *Centroid*: for each cell, the centroid is computed as the average of vectors

Scalar and vector quantization

Scalar quantization

Débit 21 bpp PSNR 47.19 dB TC 1.143

Scalar quantization

Débit 18 bpp PSNR 42.38 dB TC 1.333

Scalar quantization

Débit 15 bpp PSNR 36.97 dB TC 1.600

Scalar quantization

Débit 12 bpp PSNR 31.40 dB TC 2.000

Scalar quantization

Débit 9 bpp PSNR 29.26 dB TC 2.667

Vector quantization

Débit 9.0 bpp PSNR 37.59 TC 2.667

Scalar quantization

Débit 6 bpp PSNR 27.83 dB TC 4.000

Vector quantization

Débit 6.0 bpp PSNR 33.00 TC 4.000

Vector quantization

Débit 4.5 bpp PSNR 30.78 TC 5.333

Scalar quantization

Débit 3 bpp PSNR 25.75 dB TC 8.000

Vector quantization

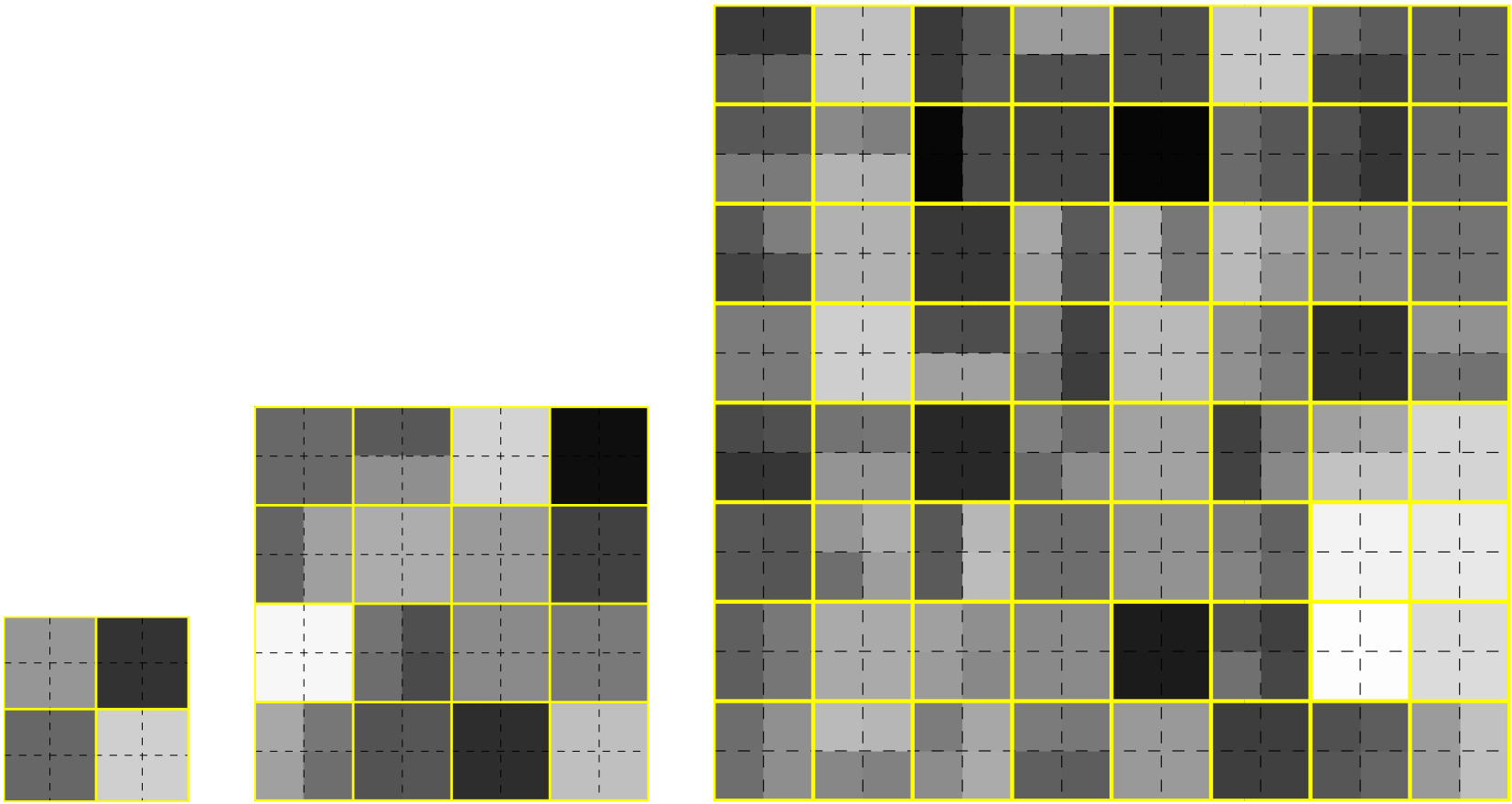
Débit 3.0 bpp PSNR 27.63 TC 8.000

Vector quantization

Débit 1.5 bpp PSNR 21.41 TC 16.000

Codebook examples

$N = 4, L \in \{4, 16, 64\}$



Outline

Performances

Bennet's formula for SQ:

$$\sigma_{\mathrm{Q}}^2 = \frac{1}{12}\left[\int_{\mathbb{R}} p_X^{1/3}(x)dx\right]^3 2^{-2b}$$

It can be generalized to VQ in N dimensions:

$$\sigma_{\mathrm{Q}}^2(N) = \alpha(N)\left[\int_{\mathbb{R}^N} p_{\underline{X}}^{\frac{N}{N+2}}(\underline{x})d\underline{x}\right]^{\frac{N+2}{N}} 2^{-2b}$$

In the Gaussian case:

$$\sigma_{\mathrm{Q}}^2(N) = \alpha(N)\left[\int_{\mathbb{R}^N}\left(\frac{\exp(\underline{x}^T\boldsymbol{R}\underline{x})}{(2\pi)^{N/2}\sqrt{\det\boldsymbol{R}}}\right)^{\frac{N}{N+2}} d\underline{x}\right]^{\frac{N+2}{N}} 2^{-2b}$$

Performances

Gaussian case

$$\sigma_Q^2(N) = \alpha(N)\left[\int_{\mathbb{R}^N}\left(\frac{\exp(\underline{x}^T\boldsymbol{R}\underline{x})}{(2\pi)^{N/2}\sqrt{\det\boldsymbol{R}}}\right)^{\frac{N}{N+2}} d\underline{x}\right]^{\frac{N+2}{N}} 2^{-2b}$$

$$= \alpha(N)2\pi\left(\frac{N+2}{N}\right)^{\frac{N+2}{2}}(\det\boldsymbol{R})^{\frac{1}{N}}2^{-2b}$$

$$= c(N)(\det\boldsymbol{R})^{\frac{1}{N}}2^{-2b}$$

It can be shown that:

$$c(1) = \frac{\sqrt{3}}{2}\pi > c(N) > c(\infty) = 1$$

Performances

Gaussian case

VQ Gain = QS MSE / QV MSE

$$\sigma_Q^2(N) = c(N)(\det \boldsymbol{R})^{\frac{1}{N}} 2^{-2b}$$

$$G_v(N) = \frac{\sigma_Q^2(1)}{\sigma_Q^2(N)} = \frac{c(1)}{c(N)} \frac{\sigma_X^2}{(det\boldsymbol{R})^{\frac{1}{N}}}$$

Geometric gain : $\frac{c(1)}{c(N)}$

Correlation gain : $\frac{\sigma_X^2}{(det\boldsymbol{R})^{\frac{1}{N}}}$

Performances

Geometric gain

- Ratio $c(1)/c(N)$ is always grater than 1
- The term $C(N)$ decreases with N
- This shows that QV is better than QS even for memoryless sources
- However, the geometric gain is limited by:

$$\lim_{N \to +\infty} \frac{c(1)}{c(N)} = c(1) = \frac{\sqrt{3}}{2}\pi$$

$$10 \log_{10} \lim_{N \to +\infty} \frac{c(1)}{c(N)} = 4.35\text{dB}$$

Performances

Geometric gain interpretation

- The geometric gain is related to the VQ's improved capability to fill $\mathbb{R}^N$ with respect to QS
- Ideal cells are hyperspheres: VQ cells tend to approximate this configuration
- QS cells are hypercubes, which are more and more different from hyperspheres when N increases

Performances

Geometric gain interpretation

- Let us consider an hypercube of dimension N
- Vertices:

Performances

Geometric gain interpretation

- Let us consider an hypercube of dimension N
- Vertices: $[\pm 1, \pm 1, \pm 1, \ldots \pm 1]$
- Faces: equation

Performances

Geometric gain interpretation

- Let us consider an hypercube of dimension N
- Vertices: $[\pm 1, \pm 1, \pm 1, \ldots \pm 1]$
- Faces: equation $x_i = \pm 1$
- Distance center-vertex:

Performances

Geometric gain interpretation

- Let us consider an hypercube of dimension N
- Vertices: $[\pm 1, \pm 1, \pm 1, \ldots \pm 1]$
- Faces: equation $x_i = \pm 1$
- Distance center-vertex: $\sqrt{N}$
- Distance center-face:

Performances

Geometric gain interpretation

- Let us consider an hypercube of dimension N
- Vertices: $[\pm 1, \pm 1, \pm 1, \ldots \pm 1]$
- Faces: equation $x_i = \pm 1$
- Distance center-vertex: $\sqrt{N}$
- Distance center-face: 1

Performances

Geometric gain interpretation

- Let us consider an hypercube of dimension N
- Vertices: $[\pm 1, \pm 1, \pm 1, \ldots \pm 1]$
- Faces: equation $x_i = \pm 1$
- Distance center-vertex: $\sqrt{N}$
- Distance center-face: 1
- When N increases hypercubes have points with a constant distance to the center and points farther and farther away from the center

Performances

Correlation gain

Correlation gain has the same expression as in the case of predictive SQ:

$$G_p = \frac{\sigma_X^2}{(\det \boldsymbol{R})^{1/N}}$$

PQ and VQ exploit the same property, i.e. sample correlation, *but* VQ can additionally benefit from geometric gain

Performances

Theoretical bound

- For a stationary signal, when N is large enough, we have $G_v > G_p$
- When N increases further, VQ performances reach an upper bound
- Any coding technique can be sees as QV
- Therefore it can be improved by GLA
- Is VQ the ultimate coding technique?

Outline

VQ in practice

Building the codebook

- Parameter selection: rate b, codebook size L, vector dimension N

$$b = \frac{\log_2 L}{N} \qquad L = 2^{bN}$$

  - The rate b is related to the target quality and to memory and transmission bounds
  - Dimension N influences performances: A rule of thumb is: take the largest N such as $x(n)$ and $x(n+N)$ still have some correlation
  - But the complexity increases exponentially with N
  - The codebook size L grows exponentially with b and N

Quantizer operation

- We have a codebook of L codewords of dimension N
- We group N input signal samples into a vector $\underline{x}$ of dimension N
  - N consecutive samples of a sound
  - N pixels of an image (typically a block)
- **Quantization**: we look into the codebook for the best representation of $\underline{x}$
  - Nearest neighbor rule
  - We compute L distances; each time this requires N multiplications: complexity $O(N)$
  - In total $O(LN) = O(N2^{bN})$

Quantizer operation

Decoding

- The encoder just sends the index of the selected codeword
- The decoder picks the codeword in the codebook
- Asymmetric process: high-complexity encoding, low-complexity decoding

Quantizer operation

Example: speech coding

- Constraints
  - Bandwidth: 8 kbps
  - Sampling frequency: 8 kHz
  - Computational capacity: $C = 1$ Gflops = 10^9 floating point operations per second
- Find N et L

Quantizer operation

Example: speech coding

- Resolution = Bits/s divides by samples/s

$$b = \frac{8\text{kb/s}}{8\text{kS/s}} = 1 b/S$$

- Operations per vector: $NL = N2^N$
- Operations per sample : 2^N
- Operations par second : $2^N f_E$
- Constraint: $2^N f_E < C$

$$2^N[\text{Op/S}] \times 8000[\text{S/s}] < 10^9[\text{Op/s}]$$

$$N < \log_2 \frac{10^9}{8 \times 10^3} = 6 \log_2 10 - 3 = 16.93$$

$$N = 16 \qquad L = 65536$$

Quantizer operation

Example: speech coding

- $N = 16$ is not very large!
- This corresponds to the fact that the speech signal is too complex to be represented with only a few tens thousands vectors (2^{16})
- The “analysis window” is too short:

$$T = N/f_E = 16/8000 = 2\text{ms}$$

- The inter-window correlation is not negligible, therefore we cannot attain the best VQ performance

Quantizer operation

Example: speech coding

- We would like to increase N (and L) without increasing too much the encoder complexity
- This is possible (by giving up some rate/distortion performance) by imposing some structure to the codebook that reduces encoding complexity

Tree-structured vector quantization (TS-VQ)

- It simplifies the NN rule
- The search of the NN is organized in levels: at each level we discard half of the candidates (binary search)
- At each step we compare the input vector with a representative of the halves of the codebook
- Then we consider only the selected half of the codebook
- This is iterated until we find a single codeword
- Complexity: from $O(2^{bN})$ to $O(bN)$
- Performance close to the *full search* case

Product vector quantization

- We decompose a vector of dimension N into k sub-vectors
- The i-th subvector has dimension N_i, such that $\sum_{i=1}^{k} N_i = N$
- Complexity : $O(\sum 2^{bN_i})$
- Example : $N = 10$, $b = 1$, $C = 2^{10} = 1024$
- $N_1 = 5$, $N_2 = 5$, $C = 2^5 + 2^5 = 64$
- We are no longer able to exploit statistical dependencies among the k subvectors
- Therefore if we process the vector in such a way to reduce these dependencies, we can effectively use this technique

Multi-stage vector quantization (MS-VQ)

- We use successive approximations
- We apply a first quantization
- Then we compute the vector quantization error
- This error is quantified
- We continue by iterating this process: residual error computation and quantization

Transform vector quantization (TVQ)

- Transform allows to concentrate information and reduce correlation
- In turn, this allows reducing N, since it should be selected as the maximum distance between correlated samples

Algebraic vector quantization AVQ

- The codebook is no longer build with GLA
- It is independent from source statistics
- Space $\mathbb{R}^N$ is partitioned in regular manner
- Advantage: large complexity reduction, no need of storing the codebook
- Disadvantage: losses in rate/distortion performance

Gain-shape vector quantization (GS-VQ)

- A QV technique popular for audio coding
- It allows to take into account power evolution along the time
- It can be seen as a special case of PVQ
- The input vector is decomposed into gain and shape, which are separately quantized

Gain-shape vector quantization

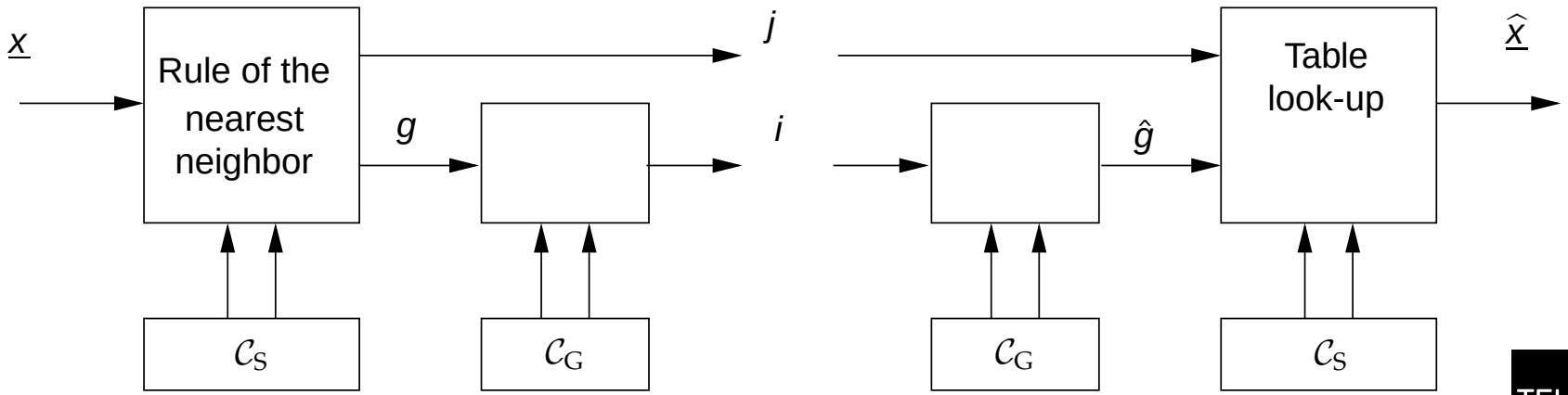
Nearest neighbor rule

- We use two codebooks
  - Shape codebook (normalized) :

$$\mathcal{C}_S = \left\{ \underline{\widehat{x}}^1, \underline{\widehat{x}}^2, \ldots, \underline{\widehat{x}}^{L_1} \right\}$$

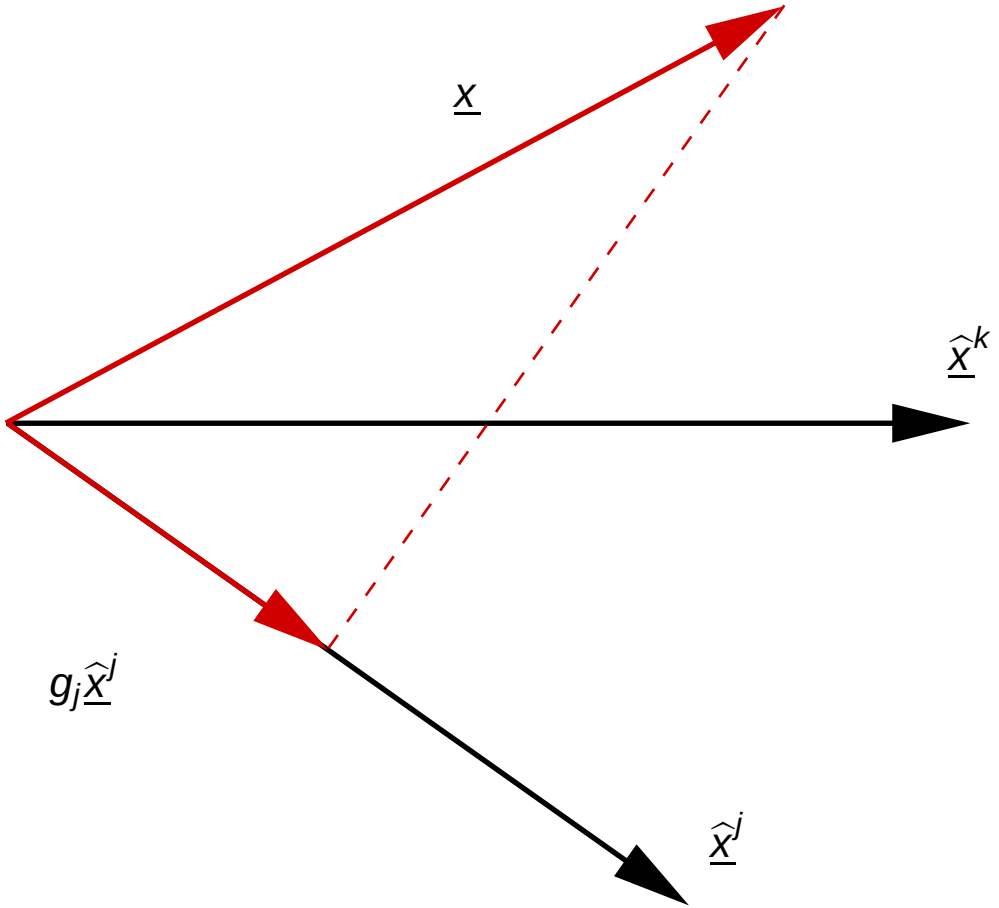
  - Gain codebook:

$$\mathcal{C}_G = \left\{ \hat{g}^1, \hat{g}^1, \ldots, \hat{g}^{L_2} \right\}$$



Gain-shape vector quantization

Nearest neighbor rule



Gain-shape vector quantization

Nearest neighbor rule

- First we look for the shape that is nearest to $\underline{x}$
- This can be done without normalization (i.e. without computing the gain):

$$\langle \underline{x} - g_j \widehat{\underline{x}}^j , \widehat{\underline{x}}^j \rangle = 0 \qquad g_j = \frac{\langle \underline{x}, \widehat{\underline{x}}^j \rangle}{\| \widehat{\underline{x}}^j \|^2}$$

$$\| \underline{x} - g_j \widehat{\underline{x}}^j \|^2 = \| \underline{x} \|^2 + g_j^2 \| \widehat{\underline{x}}^j \|^2 - 2 g_j \langle \underline{x}, \widehat{\underline{x}}^j \rangle = \| \underline{x} \|^2 - \frac{\langle \underline{x}, \widehat{\underline{x}}^j \rangle^2}{\| \widehat{\underline{x}}^j \|^2}$$

$$j^* = \arg\max_j \frac{\langle \underline{x}, \widehat{\underline{x}}^j \rangle^2}{\| \widehat{\underline{x}}^j \|^2} = \arg\max_j \left\langle \underline{x}, \frac{\widehat{\underline{x}}^j}{\| \widehat{\underline{x}}^j \|} \right\rangle^2$$

$$= \arg\max_j | \cos \phi_j |$$

Gain-shape vector quantization

GLA for GS-VQ

- Shape codebook: $\mathcal{C}_S = \left\{\underline{\widehat{x}}^1, \underline{\widehat{x}}^2, \ldots, \underline{\widehat{x}}^{L_1}\right\}$
- Normalized vectors
- Training set: M vectors $\underline{x}(m)$ with $i \in \{1, \ldots, M\}$
- GLA : Nearest neighbor and best representative
  - The best representative is no longer necessarily the centroid, since it must be normalized
- The nearest neighbor rule allows to partition vectors into cells; w.l.o.g., the j-th cell is

$$\Theta^j = \{\underline{x}(m)\}_{m \in \{1,\ldots,M(j)\}}$$

Gain-shape vector quantization

GLA for GS-VQ

- In order to optimize the representative of the j-th cell, we look for a vector $\underline{\widehat{x}}^j$ that minimizes error in the cell Θ^j, subject to the constraint of unitary norm:

$$\arg\min_{\underline{\widehat{x}}^j} \sigma_Q^2(j) \qquad s.t. \|\underline{\widehat{x}}^j\|^2 = 1$$

$$\sigma_Q^2(j) = \frac{1}{M(j)} \sum_{m=1}^{M(j)} \|\underline{x}(m) - g_j \underline{\widehat{x}}^j\|^2$$

$$\|\underline{x}(m) - g_j \underline{\widehat{x}}^j\|^2 = \|\underline{x}(m)\|^2 - \frac{\langle \underline{x}(m), \underline{\widehat{x}}^j \rangle^2}{\|\underline{\widehat{x}}^j\|^2}$$

$$= \|\underline{x}(m)\|^2 - \langle \underline{x}(m), \underline{\widehat{x}}^j \rangle^2$$

Gain-shape vector quantization

GLA for GS-VQ

- Our problem becomes

$$\arg\max_{\underline{\hat{x}}^j} Q = \arg\max_{\underline{\hat{x}}^j} \sum_{m=1}^{M(j)} \langle \underline{x}(m), \underline{\hat{x}}^j \rangle^2 \qquad s.c. \|\underline{\hat{x}}^j\|^2 = 1$$

- Γ is referred to as empiric covariance matrix

$$\Gamma = \sum_{m=1}^{M(j)} \underline{x}(m)\underline{x}(m)^T$$

- We find:

$$(\underline{\hat{x}}^j)^T \Gamma \underline{\hat{x}}^j = \sum_{m=1}^{M(j)} (\underline{\hat{x}}^j)^T \underline{x}(m)\underline{x}(m)^T \underline{\hat{x}}^j = Q$$

Gain-shape vector quantization

GLA for GS-VQ

- The problem becomes:

$$\arg\max_{\underline{\hat{x}}^j} (\underline{\hat{x}}^j)^T \Gamma \underline{\hat{x}}^j \qquad s.c. \|\underline{\hat{x}}^j\|^2 = 1$$

- Using Lagrangian multipliers method, we have to minimize

$$J = (\underline{\hat{x}}^j)^T \Gamma \underline{\hat{x}}^j - \lambda[(\underline{\hat{x}}^j)^T \underline{\hat{x}}^j - 1]$$

- We have

$$\frac{\partial J}{\partial \underline{\hat{x}}^j} = 0 \qquad \Gamma \underline{\hat{x}}^j = \lambda(\underline{\hat{x}}^j)$$

Gain-shape vector quantization

GLA for GS-VQ

- Thus $\underline{\widehat{x}}^j$ is an eigenvector of Γ, and λ is the corresponding eigenvalue
- If we multiply $\Gamma\underline{\widehat{x}}^j = \lambda(\underline{\widehat{x}}^j)$ by $(\underline{\widehat{x}}^j)^T$, we find:

$$(\underline{\widehat{x}}^j)^T \Gamma \underline{\widehat{x}}^j = \lambda (\underline{\widehat{x}}^j)^T \underline{\widehat{x}}^j$$
$$Q = \lambda$$

- Therefore, the optimization problem is solved by using the eigenvector corresponding to the largest eigenvalue of the empiric covariance matrix