

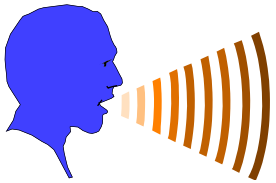
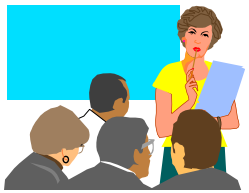



Image and Video Processing

Binary Encoding and Quantization

Yao Wang
Tandon School of Engineering, New York University

Outline

- Need for compression
- Review of probability and stochastic processes
- Entropy as measure of uncertainty and lossless coding bounds
- Huffman coding
- Arithmetic coding
- Binarization
- Scalar quantization
- Vector quantization

Necessity for Signal Compression

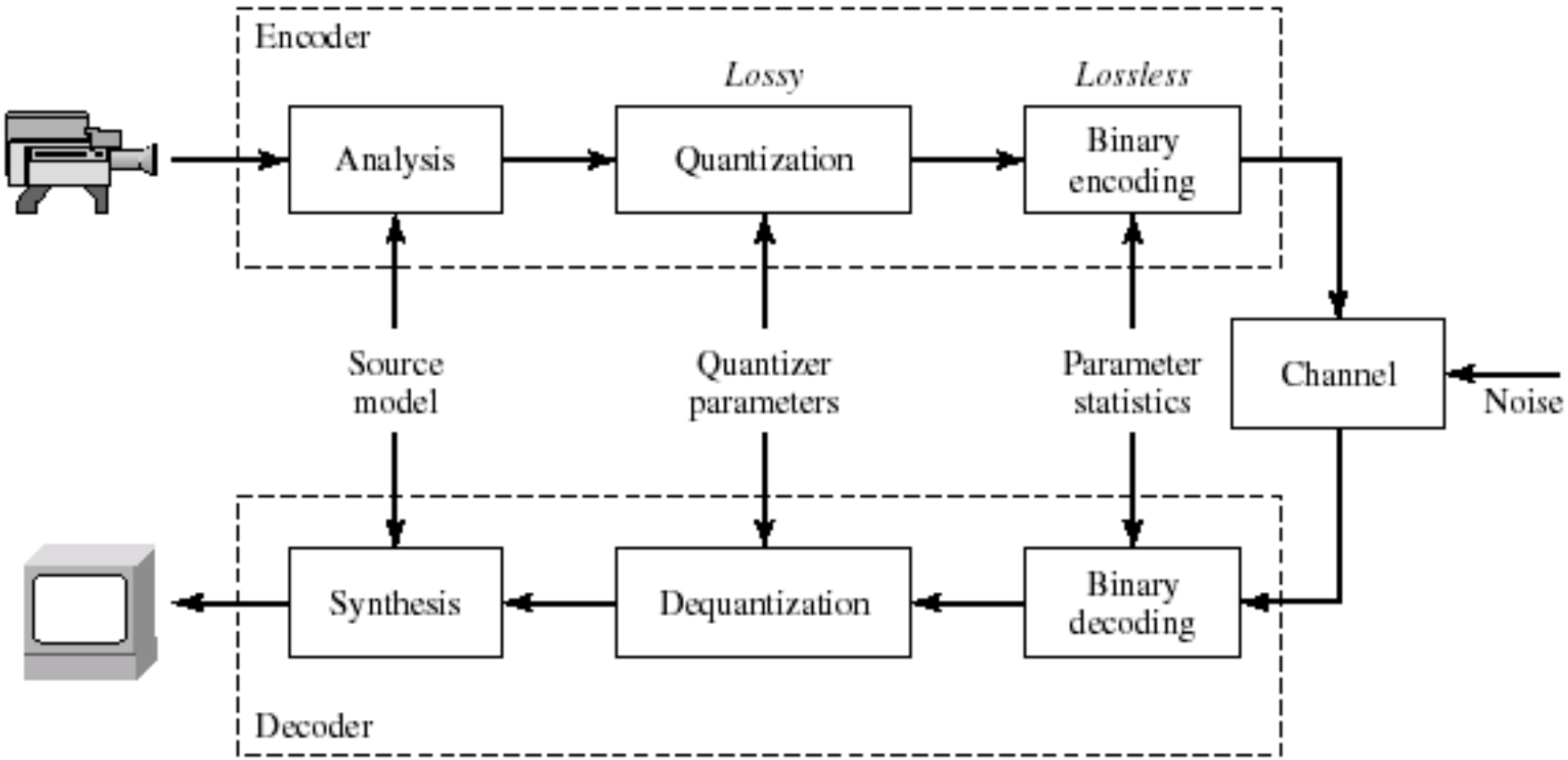
Image / Video format	**Size**
One small VGA size picture (640x480, 24-bit color)	922 KB
One large 12 MB pixel picture (3072x4096) 24-bit color still image	36 MB
Animation (320x640 pixels, 16-bit color, 16 frame/s)	6.25 MB/second
SD Video (720x480 pixels, 24-bit color, 30 frame/s)	29.7 MB/second
HD Video (1920x1080 pixels, 24-bit color, 60 frame/s)	356 MB/second

Image/Video Coding Standards by ITU and ISO

- G3,G4: facsimile standard
- JBIG: The next generation facsimile standard
  - ISO Joint Bi-level Image experts Group
- JPEG: For coding still images or video frames.
  - ISO Joint Photographic Experts Group
- JPEG2000: For coding still images, more efficient than JPEG
- Lossless JPEG: for medical and archiving applications.
- MPEGx: audio and video coding standards of ISo
- H.26x: video coding standard of ITU-T

- ITU: International telecommunications union
- ISO: International standards organization

Components in a Coding System



Binary Encoding

- Binary encoding
  - To represent a finite set of symbols using binary codewords.
- Fixed length coding
  - *N* levels represented by *(int)* $log_2(N)$ bits.
  - Ex: simple binary codes
- Variable length coding
  - more frequently appearing symbols represented by shorter codewords (Huffman, arithmetic, LZW=zip).
- The minimum number of bits required to represent a sequence of random variables is bounded by its entropy.

Reviews of Random Variables (not covered during the lecture)

- What is random variables
- A single RV
  - Pdf (continuous RV), pmf (discrete RV)
  - Mean, variance
  - Special distributions (uniform, Gaussian, Laplacian, etc.)
- Function of a random variable
- Two and multiple RV
  - Joint probability, marginal probability
  - Conditional probability
  - Conditional mean and co-variance

Examples of Random Variables

- Tossing two coins, X is the number of heads, and Y is the number of tails
  - X and Y take on values {0, 1, 2}
  - Discrete type
- X is the lifetime of a certain brand of light bulbs
  - X take on values [0, +∞)
  - Continuous type

Distribution, Density, and Mass Functions

- The cumulative distribution function (cdf) of a random variable X, is defined by $$F_X(x) = \Pr.(X \le x), \text{for all x.}$$
- If X is a continuous random variable (taking value over a continuous range)
  - $F_X(x)$ is continuous function.
  - The probability density function (pdf) of X is given by $$f_X(x) = \frac{d}{dx} F_X(x)$$
- If X is a discrete random variable (taking a finite number of possible values)
  - $F_X(x)$ is step function.
  - The probability mass function (pmf) of X is given by

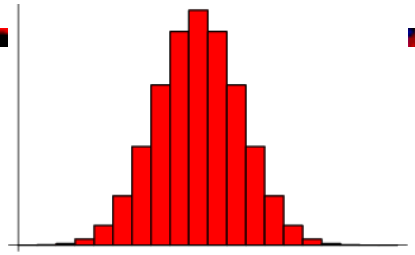
$$p_X(x) = \Pr.(X = x)$$

The percentage of time that X=x.

Special Cases

- Binomial (discrete)

$$P\{X=k\}=\binom{n}{k}p^k(1-p)^{n-k}, k=0,1,\ldots,n.$$



- Poisson distribution

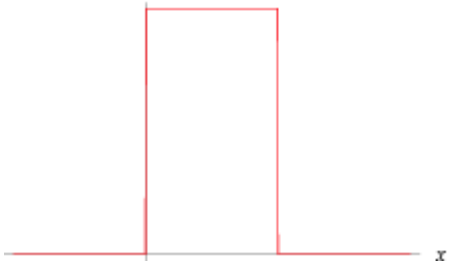
$$P\{X=k\}=e^{-a}\frac{a^k}{k!}, k=0,1,\ldots$$

- Normal (or Gaussian) $N(\mu, \sigma^2)$

$$f(x)=\frac{1}{\sqrt{2\pi}\sigma}e^{-(x-\mu)^2/(2\sigma^2)}$$

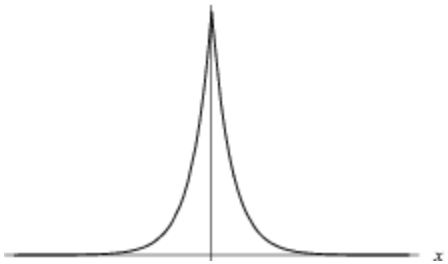
- Uniform over (x_1, x_2),

$$f(x)=\begin{cases}\frac{1}{x_2-x_1} & x_1\le x\le x_2\\ 0 & \text{otherwise}\end{cases}$$



- Laplacian $L(\mu, b)$

$$f(x)=\frac{1}{2b}e^{-|x-\mu|/b}$$



Figures are from http://mathworld.wolfram.com

Expected Values

- The expected (or mean) value of a random variable X:

$$\eta_X = E\{X\} = \begin{cases} \int_{-\infty}^{\infty} x f_X(x)dx & \text{if X is continuous} \\ \sum_{x\in X} xP(X=x) & \text{if X is discrete} \end{cases}$$

- The variance of a random variable X:

$$\sigma_X^2 = Var\{X\} = \begin{cases} \int_{-\infty}^{\infty} (x-\eta_X)^2 f_X(x)dx & \text{if X is continuous} \\ \sum_{x\in X} (x-\eta_X)^2 P(X=x) & \text{if X is discrete} \end{cases}$$

- Mean and variance of common distributions:
  - Uniform over range (x_1, x_2): $E\{x\} = (x_1+x_2)/2$, $VarX = (x_2-x_1)^2/12$
  - Gaussian $N(\mu, \sigma^2)$: $Ex = \mu$, $VarX = \sigma^2$
  - Laplace $L(\mu, b)$: $Ex = \mu$, $VarX = 2b^2$

Functions of Random Variable

- Y=g(X)
  - Following the example of the lifetime of the bulb, let Y represents the cost of a bulb, which depends on its lifetime X with relation $Y = \sqrt{X}$

- Expectation of Y

$$\eta_Y = E\{Y\} = \begin{cases} \int_{-\infty}^{\infty} g(x) f_X(x) dx & \text{if X is continuous} \\ \sum_{x \in X} g(x) P(X = x) & \text{if X is discrete} \end{cases}$$

- Variance of Y

$$\sigma_Y^2 = Var\{Y\} = \begin{cases} \int_{-\infty}^{\infty} (g(x) - \eta_Y)^2 f_X(x) dx & \text{if X is continuous} \\ \sum_{x \in X} (g(x) - \eta_Y)^2 P(X = x) & \text{if X is discrete} \end{cases}$$

Two RVs

- We only discuss discrete RVs (i.e. X and Y for both discrete RVs)
- The joint probability mass function (pmf) of X and Y is given by

$$p_{XY}(x,y) = \Pr.(X = x, Y = y)$$

- The conditional probability mass function of X given Y is

$$p_{X/Y}(x/y) = \Pr.(X = x \mid Y = y)$$

- Important relations

$$p_{XY}(x,y) = p_{X/Y}(x/y)p_Y(y)$$

$$p_X(x) = \sum_{y \in Y} \Pr._{XY}(X = x, Y = y)$$

Conditional Mean and Covariance

- Conditional mean

$$\eta_{X|y} = E\{X \mid y\} = \sum_{x \in \mathrm{X}} xP(X = x \mid Y = y)$$

- Correlation

$$R_{X,Y} = E\{XY\} = \sum_{x \in \mathrm{X}, y \in \mathrm{Y}} xyP(X = x, Y = y)$$

- Correlation matrix

$$\mathbf{R} = E\left\{\begin{bmatrix} X \\ Y \end{bmatrix}\begin{bmatrix} X & Y \end{bmatrix}\right\} = \begin{bmatrix} E\{X^2\} & R_{XY} \\ R_{XY} & E\{Y^2\} \end{bmatrix}, \quad E\{X^2\} = \sigma_X^2 + \eta_X^2$$

- Covariance

$$C_{X,Y} = E\{(X - \eta_X)(Y - \eta_Y)\} = R_{X,Y} - \eta_X \eta_Y$$

- Covariance matrix

$$\mathbf{C} = E\left\{\begin{bmatrix} X - \eta_X \\ Y - \eta_Y \end{bmatrix}\begin{bmatrix} X - \eta_X & Y - \eta_Y \end{bmatrix}\right\} = \begin{bmatrix} \sigma_X^2 & C_{XY} \\ C_{XY} & \sigma_Y^2 \end{bmatrix}$$

Multiple RVs

- The definitions for two RVs can be easily extended to multiple (N>2) RVs, $X_1, X_2, \ldots, X_N$
- The joint probability mass function (pmf) is given by

$$p(x_1, x_2, \ldots, x_N) = \Pr.(X_1 = x_1, X_2 = x_2, \ldots, X_N = x_N)$$

- Covariance matrix is

$$\mathbf{C} = E\left\{\begin{bmatrix} X_1 - \eta_1 \\ X_2 - \eta_2 \\ \ldots \\ X_N - \eta_N \end{bmatrix} \begin{bmatrix} X_1 - \eta_1 & X_2 - \eta_2 & \ldots & X_N - \eta_N \end{bmatrix}\right\} = \begin{bmatrix} \sigma_1^2 & C_{12} & \ldots & C_{1N} \\ C_{21} & \sigma_2^2 & & C_{2N} \\ \ldots & \ldots & \ldots & \ldots \\ C_{N1} & C_{N2} & \ldots & \sigma_N^2 \end{bmatrix}$$

Statistical Characterization of Random Sequences

- Random sequence (a discrete time random process)
  - Ex 1: an image that follows a certain statistics $\mathcal{F} = \{\mathcal{F}_n\}$
    - F_n represents the *possible value* of the n-th pixel of the image, ***n****=(m,n)*
    - f_n represents the *actual value* taken
  - Ex 2: a video that follows a certain statistics
    - F_n represents the *possible value* of the n-th pixel of a video, ***n****=(k,m,n)*
    - f_n represents the actual value taken
  - Continuous source*:* F_n takes continuous values (analog image)
  - Discrete source: F_n takes discrete values (digital image)
- Stationary source: statistical distribution invariant to time (space) shift
- Probability distribution
  - probability mass function (pmf) or probability density function (pdf): $p_{\mathcal{F}_n}(f)$ $p(f)$
  - Joint pmf or pdf: $p_{\mathcal{F}_{n+1},\mathcal{F}_{n+2},\ldots,\mathcal{F}_{n+N}}(f_1, f_2, \ldots, f_N)$ $p(f_1, f_2, \ldots f_N)$
  - Conditional pmf or pdf:

$$p_{\mathcal{F}_n|\mathcal{F}_{n-1},\mathcal{F}_{n-2},\ldots,\mathcal{F}_{n-M}}(f_{M+1} \mid f_M, f_{M-1}, \ldots, f_1) \qquad p(f_{M+1} \mid f_M, f_{M-1}, \ldots, f_1)$$

Entropy and Mutual Information

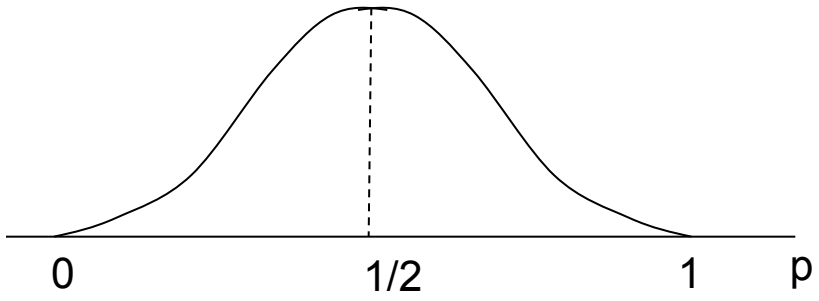
- Single RV: entropy
- Multiple RV: joint entropy, conditional entropy, mutual information

Entropy of a RV

- Consider RV $F=\{f_1,f_2,\ldots,f_K\}$, with probability $p_k=Prob.\{F=f_K\}$
- Self-Information of one realization f_k : $H_k= -\log(p_k)$
  - p_k=1: always happen, no information
  - P_k ~0: seldom happen, its realization carries a lot of information
- Entropy = average information: $$H(\mathcal{F}) = -\sum_{f\in\mathcal{A}} p_{\mathcal{F}}(f)\log_2 p_{\mathcal{F}}(f).$$
  - Entropy is a measure of uncertainty or information content, unit=bits
  - Very uncertain -> high information content

Example: Two Possible Symbols

- Example: Two possible outcomes
  - Flip a coin, F={“head”,”tail”}: $p_1=p_2=1/2$: H=1 (highest uncertainty)
  - If the coin has defect, so that $p_1=1$, $p_2=0$: H=0 (no uncertainty)
  - More generally: $p_1=p$, $p_2=1-p$,
    - H=-(p log p+ (1-p) log (1-p))
    - H is maximum when p=1/2 (most uncertain)



Another Example: English Letters

- 26 letters, each has a certain probability of occurrence
  - Some letters occurs more often: “a”,”s”,”t”, …
  - Some letters occurs less often: “q”,”z”, …
- Entropy ~= information you obtained after reading an article.
- But we actually don’t get information at the alphabet level, but at the word level!
  - Some combination of letters occur more often: “it”, “qu”,…

Joint Entropy

- Joint entropy of two RVs:
  - Uncertainty of two RVs together

$$H(\mathcal{F},\mathcal{G}) = -\sum_{f\in\mathcal{A}_f}\sum_{g\in\mathcal{A}_g} p_{\mathcal{F},\mathcal{G}}(f,g)\log_2 p_{\mathcal{F},\mathcal{G}}(f,g).$$

$$H(\mathcal{F},\mathcal{G}) \le H(\mathcal{F}) + H(\mathcal{G})$$

- N-th order entropy
  - Uncertainty of N successive samples of a random sequence

$$\begin{aligned} H_N(\mathcal{F}) &= H(\mathcal{F}_1,\mathcal{F}_2,\ldots,\mathcal{F}_N) \\ &= -\sum_{[f_1,f_2,\ldots,f_N]\in\mathcal{A}^N} p(f_1,f_2,\ldots,f_N)\log_2 p(f_1,f_2,\ldots,f_N). \end{aligned}$$

- Entropy rate (lossless coding bound)
  - Average uncertain per RV

$$\bar{H}(\mathcal{F}) = \lim_{N\to\infty}\frac{1}{N}H_N(\mathcal{F}) = \lim_{N\to\infty} H_{C,N}(\mathcal{F}).$$

Conditional Entropy

- Conditional entropy between two RVs:
  - Uncertainty of one RV given the other RV

$$H(\mathcal{F}\,|\,\mathcal{G}) = \sum_{g\in\mathcal{A}_g} p_{\mathcal{G}}(g)\,H(\mathcal{F}\,|\,g)$$

$$= -\sum_{g\in\mathcal{A}_g} p_{\mathcal{G}}(g) \sum_{f\in\mathcal{A}_f} p_{\mathcal{F}|\mathcal{G}}(f\,|\,g)\log_2 p_{\mathcal{F}|\mathcal{G}}(f\,|\,g).$$

$$H(\mathcal{F}) \geq H(\mathcal{F}\,|\,\mathcal{G}) \qquad H(\mathcal{F},\mathcal{G}) = H(\mathcal{G}) + H(\mathcal{F}\,|\,\mathcal{G})$$

- M-th order conditional entropy

$$H_{C,M}(\mathcal{F}) = H(\mathcal{F}_{M+1}\,|\,\mathcal{F}_M,\mathcal{F}_{M-1},\ldots,\mathcal{F}_1)$$

$$= \sum_{[f_1,f_2,\ldots,f_M]\in\mathcal{A}^M} p(f_1,f_2,\ldots,f_M)\,H(\mathcal{F}_{M+1}\,|\,f_M,f_{M-1},\ldots,f_1)$$

$$H(\mathcal{F}_{M+1}\,|\,f_M,f_{M-1},\ldots,f_1)$$

$$= -\sum_{f_{M+1}\in\mathcal{A}} p(f_{M+1}\,|\,f_M,f_{M-1},\ldots,f_1)\log_2 p(f_{M+1}\,|\,f_M,f_{M-1},\ldots,f_1).$$

$$H(\mathcal{F}) \leq H_{C,N-1}(\mathcal{F}) \leq \frac{1}{N}H_N(\mathcal{F}) \leq H_1(\mathcal{F}).$$

Example: 4-symbol source

- Four symbols: “a”,”b”,”c”,”d”
- pmf:

$$\mathbf{p}^T = [0.5000, 0.2143, 0.1703, 0.1154]$$

- 1st order conditional pmf: $q_{ij}=Prob(f_i|f_j)$

$$[\mathbf{Q}] = \begin{bmatrix} 0.6250 & 0.3750 & 0.3750 & 0.3750 \\ 0.1875 & 0.3125 & 0.1875 & 0.1875 \\ 0.1250 & 0.1875 & 0.3125 & 0.1250 \\ 0.0625 & 0.1250 & 0.1250 & 0.3125 \end{bmatrix}$$

- 2nd order pmf: $p(f_{n-1}, f_n) = p(f_{n-1})q(f_n \mid f_{n-1})$.

$$\text{Ex.} \quad p(\text{"}ab\text{"}) = p(\text{"}a\text{"})q(\text{"}b\text{"}/\text{"}a\text{"}) = 0.5 * 0.1875 = 0.0938$$

- Go through how to compute H_1, H_2, $H_{c,1}$.

Mutual Information

- Mutual information between two RVs :
  - Information provided by *G* about *F*

$$I(\mathcal{F};\mathcal{G}) = \sum_{f\in\mathcal{A}_f}\sum_{g\in\mathcal{A}_g} p_{\mathcal{F},\mathcal{G}}(f,g)\log_2\frac{p_{\mathcal{F},\mathcal{G}}(f,g)}{p_{\mathcal{F}}(f)p_{\mathcal{G}}(g)}.$$

$$I(\mathcal{F};\mathcal{G}) = H(\mathcal{F}) - H(\mathcal{F}\,|\,\mathcal{G})$$

$$I(\mathcal{F};\mathcal{G}) \le H(\mathcal{F})$$

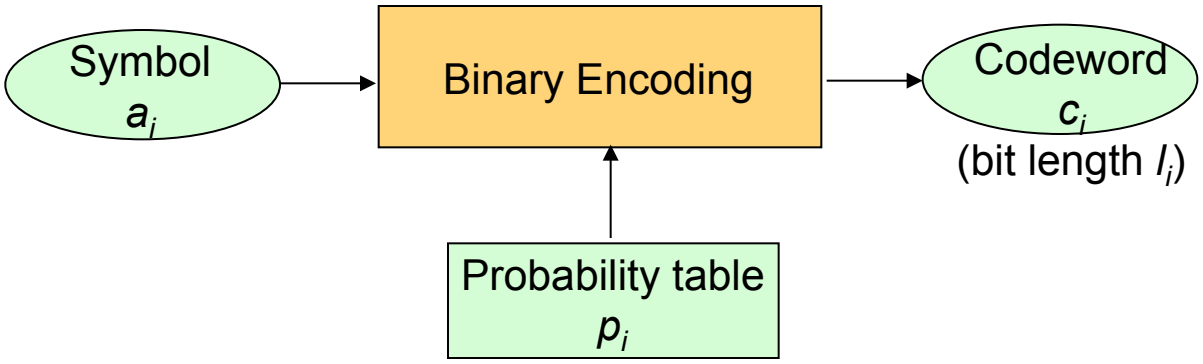
$$I(\mathcal{F};\mathcal{G}) = H(\mathcal{F}) + H(\mathcal{G}) - H(\mathcal{F},\mathcal{G})$$

- N-th order mutual information (lossy coding bound)

$$I_N(\mathcal{F};\mathcal{G}) = \sum_{[f_1,f_2,\ldots,f_N]\in\mathcal{A}_f^N}\ \sum_{[g_1,g_2,\ldots,g_N]\in\mathcal{A}_g^N} p(f_1,f_2,\ldots,f_N,g_1,g_2,\ldots,g_N)\cdot\log_2\frac{p(f_1,f_2,\ldots,f_N,g_1,g_2,\ldots,g_N)}{p(f_1,f_2,\ldots,f_N)p(g_1,g_2,\ldots,g_N)}$$

Lossless Coding (Binary Encoding)

- Binary encoding is a necessary step in any coding system
  - Applies to
    - original symbols (e.g. image pixels) in a discrete source,
    - or converted symbols (e.g. quantized transformed coefficients) from a continuous or discrete source
- Binary encoding process (scalar coding)



Bit rate (bit/symbol): $$R = \sum_{a_i \in \mathcal{A}} p(a_i) l(a_i).$$

Bound for Lossless Coding

- Scalar coding:
  - Assign one codeword to one symbol at a time
  - Problem: could differ from the entropy by up to 1 bit/symbol

$$H_1(\mathcal{F}) \le \bar{R}_1(\mathcal{F}) \le H_1(\mathcal{F}) + 1.$$

- Vector coding:
  - Assign one codeword for each group of N symbols
  - Larger N -> Lower Rate, but higher complexity

$R_N(F)$: bits for N symbols

$\bar{R}_N(F) = R_N(F)/N$: bits per symbol

$$H_N(\mathcal{F}) \le R^N(\mathcal{F}) \le H_N(\mathcal{F}) + 1 \qquad H_N(\mathcal{F})/N \le \bar{R}_N(\mathcal{F}) \le H_N(\mathcal{F})/N + 1/N.$$

$$\lim_{N\to\infty} \bar{R}_N(\mathcal{F}) = \bar{H}(\mathcal{F}).$$

- Conditional coding (context-based coding)
  - The codeword for the current symbol depends on the pattern (context) formed by the previous M symbols

$$H_{C,M}(\mathcal{F}) \le \bar{R}_{C,M}(\mathcal{F}) \le H_{C,M}(\mathcal{F}) + 1.$$

$$\bar{H}(\mathcal{F}) \le \lim_{M\to\infty} R_{C,M}(\mathcal{F}) \le \bar{H}(\mathcal{F}) + 1.$$

Binary Encoding: Requirement

- A good code should be:
  - Uniquely decodable
  - Instantaneously decodable – prefix code (aka prefix-free code)

Codebook 1
(a prefix code)

Symbol	Codeword
a_1	"0"
a_2	"10"
a_3	"110"
a_4	"111"

Codebook 2
(not a prefix code)

Symbol	Codeword
a_1	"0"
a_2	"01"
a_3	"100"
a_4	"011"

Bitstream: 0 0 1 1 0 1 0 1 1 0 1 0 0

Decoded string based on codebook 1: (can decode instantaneously)
0|0|1 1 0|1 0|1 1 0|1 0|0 → $a_1\ a_1\ a_3\ a_2\ a_3\ a_2\ a_1$

Decoded string based on codebook 2: (must look ahead to decode)
0|0 1 1|0 1|0 1 1|0|1 0 0 → $a_1\ a_4\ a_2\ a_4\ a_1\ a_3$

Huffman Coding

- Idea: more frequent symbols -> shorter codewords
- Algorithm:

Step 1: Arrange the symbol probabilities $p(a_l)$, $l = 1, 2, \ldots, L$, in a decreasing order and consider them as leaf nodes of a tree.

Step 2: While there is more than one node:

(a) Find the two nodes with the smallest probability and arbitrarily assign 1 and 0 to these two nodes.

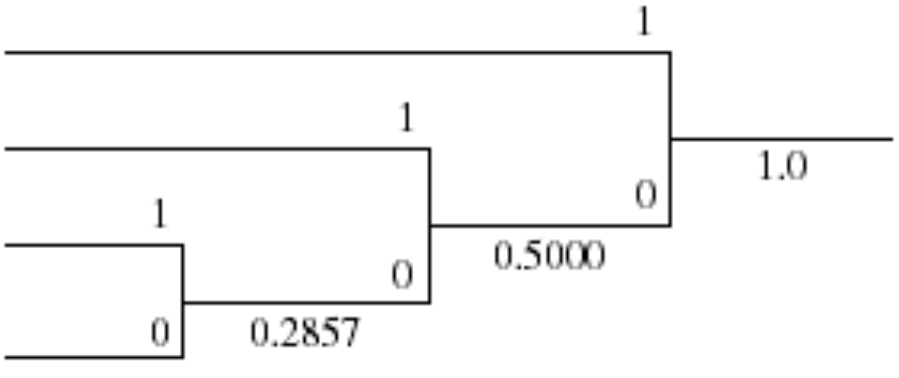
(b) Merge the two nodes to form a new node whose probability is the sum of the two merged nodes. Go back to Step 1.

Step 3: For each symbol, determine its codeword by tracing the assigned bits from the corresponding leaf node to the top of the tree. The bit at the leaf node is the last bit of the codeword.

- Huffman coding generate prefix code ☺
- Can be applied to one symbol at a time (scalar coding), or a group of symbols (vector coding), or one symbol conditioned on previous symbols (conditional coding)

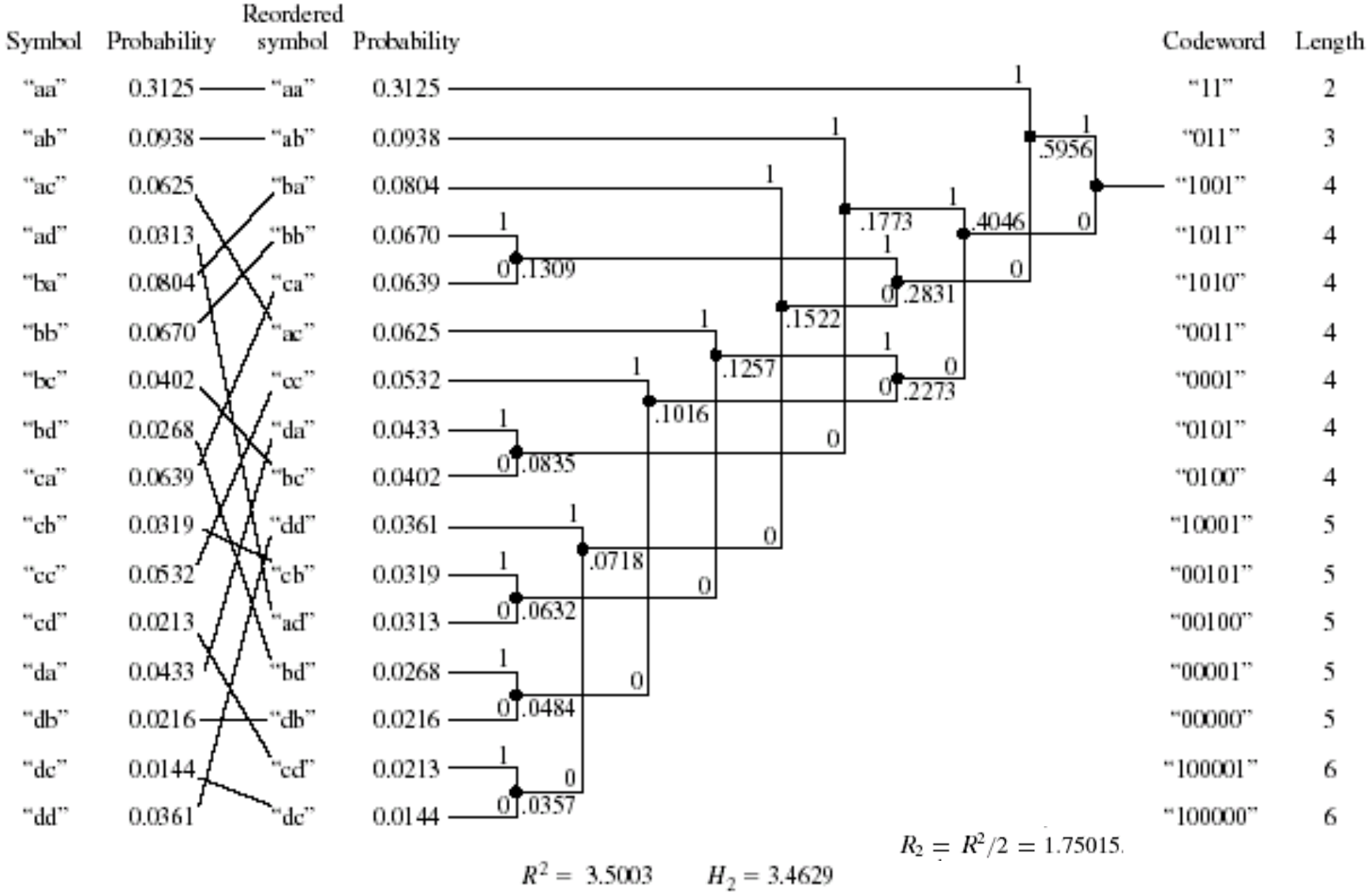
Huffman Coding Example: Scalar Coding

Symbol	Probability	Codeword	Codeword length
"a"	0.5000	"1"	1
"b"	0.2143	"01"	2
"c"	0.1703	"001"	3
"d"	0.1154	"000"	3



Bit rate $R = 1.7857$ Entropy $H_1 = 1.7707$

Huffman Coding Example: Vector Coding

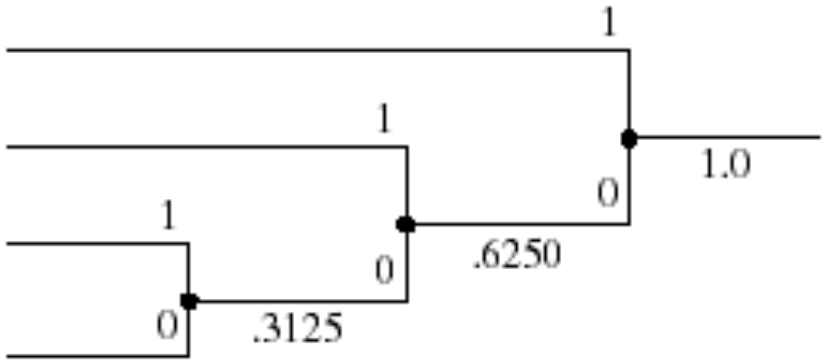


$R_2 = R^2/2 = 1.75015$

$R^2 = 3.5003 \qquad H_2 = 3.4629$

Huffman Coding Example: Conditional Coding

Symbol	Probability	Codeword	Length
"a"/"b"	0.3750	"1"	1
"b"/"b"	0.3125	"01"	2
"c"/"b"	0.1875	"001"	3
"d"/"b"	0.1250	"000"	3



$R_{C,"b"} = 1.9375 \qquad H_{C,"b"} = 1.8829$

$$R_{C,"a"} = 1.5625, R_{C,"b"} = R_{C,"c"} = R_{C,"d"} = 1.9375, R_{C,1} = 1.7500$$

$$H_{C,"a"} = 1.5016, H_{C,"b"} = H_{C,"c"} = H_{C,"d"} = 1.8829, H_{C,1} = 1.6922$$

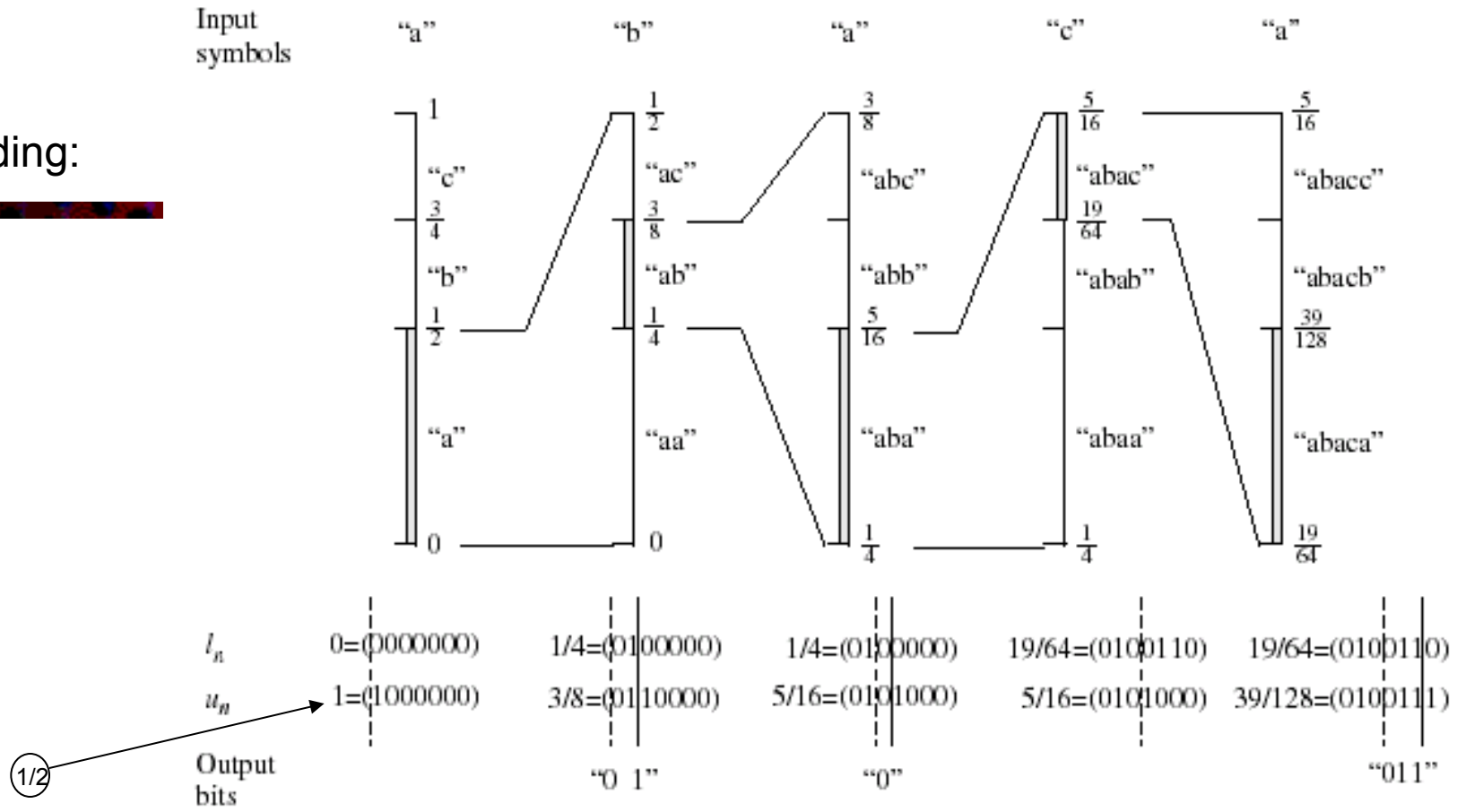
Arithmetic Coding (Not Required)

- Basic idea:
  - Represent a sequence of symbols by an interval with length d equal to its probability p
  - The interval is specified by its lower boundary (*l*), upper boundary (*u*) and length *d* (=probability)
  - The codeword for the sequence is the common bits in binary representations of *l* and *u*
  - *Theoretically, no. bits (B) = ceiling(-log$_2$ d)=ceiling (- log$_2$ p)*
  - *A more likely sequence=a longer interval=fewer bits*
- The interval is calculated sequentially starting from the first symbol
  - The initial interval is determined by the first symbol
  - The next interval is a subinterval of the previous one, determined by the next symbol

$$d_n = d_{n-1} * p_l; \quad l_n = l_{n-1} + d_{n-1} * q_{l-1}; \quad u_n = l_n + d_n.$$

P(a)=1/2
P(b)=1/4
P(c)=1/4

Encoding:



(a)

Decoding:

Received bits	Interval	Decoded symbol
"0"	[0,1/2)	"a"
"01"	[1/4,1/2)	—
"010"	[1/4,3/8)	"b"
"0100"	[1/4,5/16)	"a"
"01001"	[9/32,5/16)	—
"010011"	[19/64,5/16)	"c"
…	…	…

(b)

Implementation of Arithmetic Coding

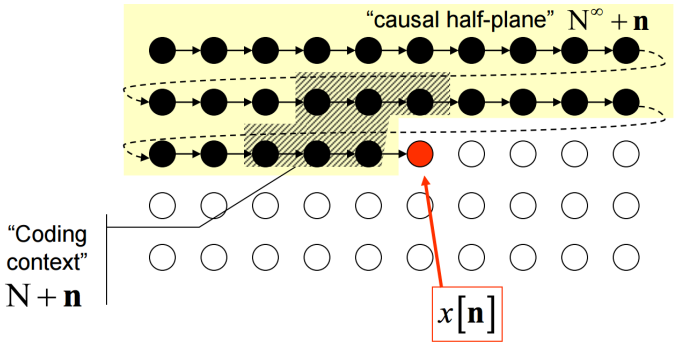
- Previous example is meant to illustrate the algorithm in a conceptual level
  - Require infinite precision arithmetic
  - Can be implemented with finite precision or integer precision only
  - Efficient implementation for coding binary symbols
- For more details on implementation, see
  - Witten, Neal and Cleary, "Arithmetic coding for data compression", J. ACM (1987), 30:520-40
  - Sayood, *Introduction to Data Compression*, Morgan Kaufmann, 1996

Binary Arithmetic Coding

- Only two possible input symbols: MPS (More probably symbol, p_m) and LPS (less probable symbol, p_l=1-p_m)
- Recursively split an interval to 2
- Simplified implementation
  - Instead of using exact probability, consider a finite predetermined set. Quantize the actual probability into one of those in the set.
  - Instead of using multiplication to calculate the new interval length, use table look up.

Context Based Binary Arithmetic Coding (CABAC)

- Instead of using the probability of the current binary symbol, use the conditional probability, conditioned on its context
- When coding a 2D binary image, the context can be the previously coded pixels in a causal neighborhood. If the context includes N pixels, there will be 2^N possible contexts. Use a look up table to store p_m or p_l of each context.
- The probability under each context is recursively updated after coding each new symbol



http://web.stanford.edu/class/ee398a/handouts/lectures/03-ArithmeticCoding.pdf

What if the source symbols are not binary?

- First represent each symbol using binary bits (binarization)
- Then apply BAC to the sequence of binarized bits
- We may use different probability for the binary bits based on their positions.

Simple Binarization

- When all symbols are equally likely
- *Simple binary code*: N possible values represented by $[\log_2 N]$ bits ([ ] represents "ceiling")
- *Truncated binary code*: use on average less than $[\log_2 N]$ when N is not power of 2
  - $2^k < N < 2^{k+1}$, $U=2^{k+1}-N$
  - First U symbols coded using k bits, remaining N-U symbols using k+1 bits

Truncated Binary Coding Example (N=5)

Truncated binary	Encoding			Standard binary
0	~~0~~	0	0	0
1	~~0~~	0	1	1
2	~~0~~	1	0	2
UNUSED	~~0~~	~~1~~	~~1~~	3
UNUSED	~~1~~	~~0~~	~~0~~	4
UNUSED	~~1~~	~~0~~	~~1~~	5/UNUSED
3	1	1	0	6/UNUSED
4	1	1	1	7/UNUSED

From: http://en.wikipedia.org/wiki/Truncated_binary_encoding

Unary Coding

n (non-negative)	n (strictly positive)	Unary code	Alternative
0	1	0	1
1	2	10	01
2	3	110	001
3	4	1110	0001
4	5	11110	00001
5	6	111110	000001
6	7	1111110	0000001
7	8	11111110	00000001
8	9	111111110	000000001
9	10	1111111110	0000000001

http://en.wikipedia.org/wiki/Unary_coding

- Unary coding is optimal for probability distribution $P(n)=2^{-n}$, n=1,2,..
- When the actual symbol does not following this distribution, to further reduce the bit rate, we can apply BAC to the sequences of bits, with probability depending on the position of the bit in a symbol. In this case, we are using the position as the context of CABAC.

Example: Unary Code + BAC

- Input sequence: {1,3, 5, 1, …}
- Binarization: 0,110,11110, 0, …
- P1=probability of “0” in the first bin
- P2=probability of “0” in the second bin
- …

- BAC(0,P1),BAC(1,P1), BAC(1,P2),BAC(0,P3), BAC(1,P1), …

Golomb-Rice Coding

- Useful for the possible number of symbols is large and smaller numbers are more likely
- Divide all possible symbols into groups of M symbols, represent a symbol by its group number (quotient) and its position in the group (remainder).
- N= q M + r
- Represent q using unary code (followed by BAC)
- Represent r using simple binary (if M=power of 2) or truncated binary

Huffman vs. Arithmetic Coding

- Huffman coding (assuming vector coding of N symbols together)
  - Convert a fixed number of N symbols into a variable length codeword
  - Efficiency: $H_N(\mathcal{F})/N \le \bar{R}_N(\mathcal{F}) \le H_N(\mathcal{F})/N + 1/N.$
  - To approach entropy rate, must code a large number of symbols together
  - Used in all earlier image and video coding standards

- Arithmetic coding
  - Convert a variable number of symbols into a variable length codeword
  - Efficiency: $H_N(\mathcal{F})/N \le R \le H_N(\mathcal{F})/N + 2/N,$ *N* is sequence length
  - Can approach the entropy rate by processing one symbol at a time
  - Easy to adapt to changes in source statistics
  - Integer implementation is available, but still more complex than Huffman coding with a small N
  - Used as advanced options in earlier image and video coding standards (JPEG, H264 and before)
  - Standard options in newer standards (JPEG2000, HEVC)

LZW coding (Not Required)

- LZW coding (Lempel, Ziv, and Welsh)
  - Assign fixed-length codewords to variable length sequences of source symbols
  - Does not require priori knowledge of the symbol probabilities. (universal code)
  - Not as efficient as Huffman for a given distribution

Summary on Binary Coding

- Coding system:
  - original data -> model parameters -> quantization-> binary encoding
  - Waveform-based vs. content-dependent coding
- Characterization of information content by entropy
  - Entropy, Joint entropy, conditional entropy
  - Mutual information
- Lossless coding
  - Bit rate bounded by entropy rate of the source
  - Huffman coding:
    - Scalar, vector, conditional coding
    - can achieve the bound only if a large number of symbols are coded together
    - Huffman coding generates prefix code (instantaneously decodable)
  - Arithmetic coding
    - Can achieve the bound by processing one symbol at a time
    - More complicated than scalar or short vector Huffman coding

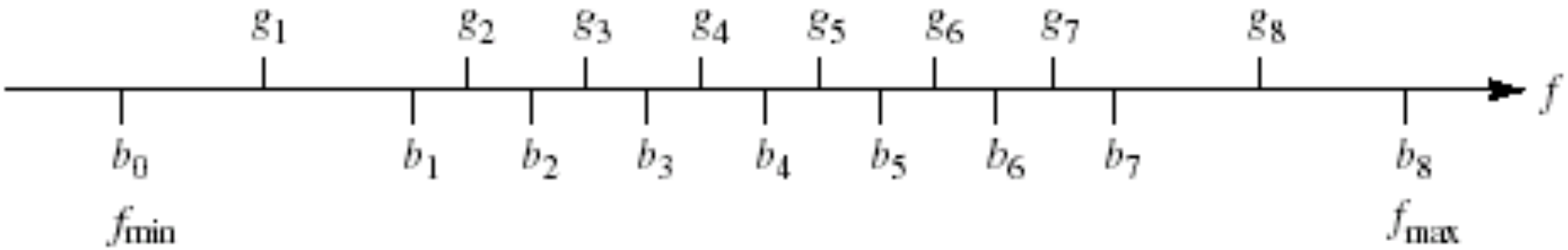
Lossy Coding

- Original source is discrete
  - Lossless coding: bit rate >= entropy rate
  - One can further quantize source samples to reach a lower rate
- Original source is continuous
  - Lossless coding will require an infinite bit rate!
  - One must quantize source samples to reach a finite bit rate
  - Lossy coding rate is bounded by the mutual information between the original source and the quantized source that satisfy a distortion criterion
- Quantization methods
    - Scalar quantization
    - Vector quantization

Scalar Quantization

- General description
- Uniform quantization
- MMSE quantizer
- Lloyd algorithm

SQ as Line Partition



Quantization levels: L

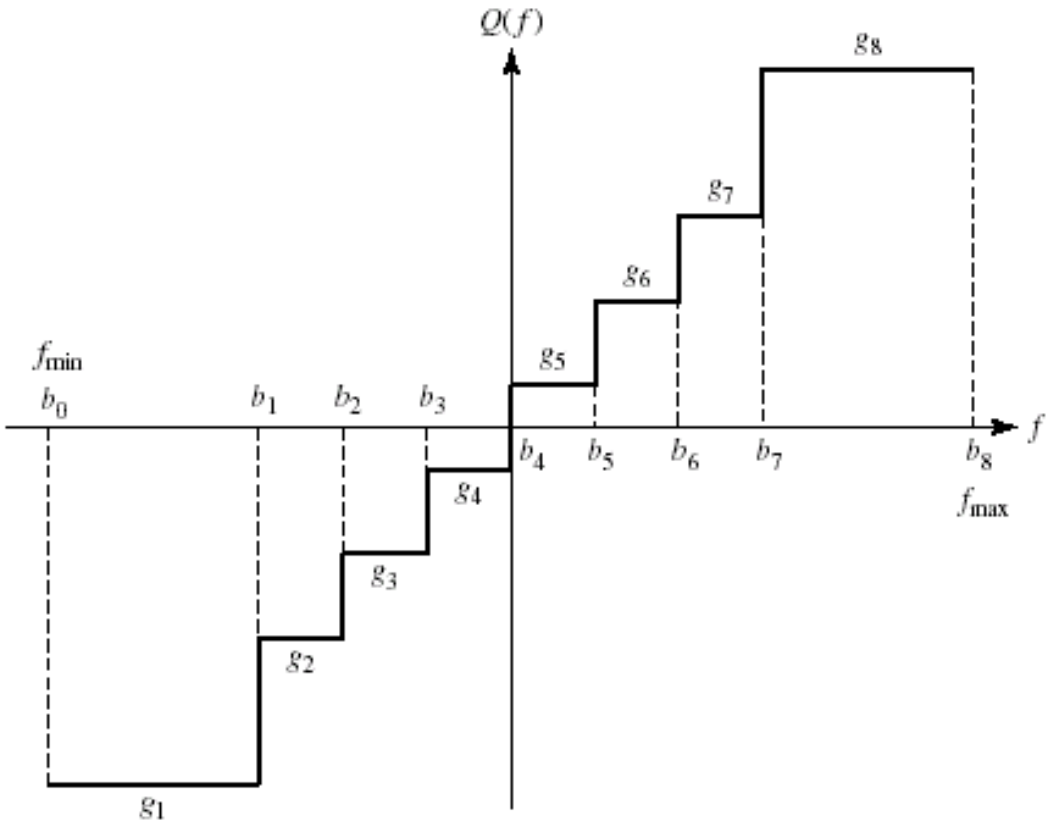
Boundary values: b_l

Partition regions: $B_l = [b_{l-1}, b_l)$

Reconstruction values: g_l

Quantizer mapping: $Q(f) = g_l$, if $f \in B_l$

Function Representation



$$Q(f) = g_l, \text{ if } f \in B_l$$

Distortion Measure

General measure:

$$D_q = E\{d_1(\mathcal{F}, Q(\mathcal{F}))\} = \int_{f\in\mathcal{B}} d_1(f, Q(f))\, p(f)\, df$$

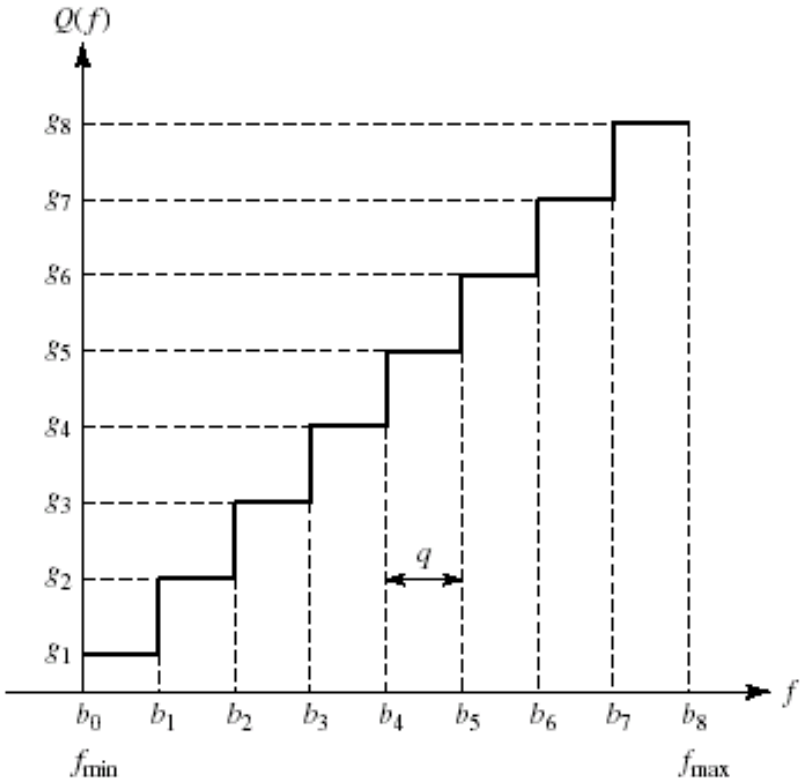
$$= \sum_{l\in\mathcal{L}} P(\mathcal{B}_l)\, D_{q,l}$$

$$D_{q,l} = \int_{f\in\mathcal{B}_l} d_1(f, g_l)\, p(f \mid f \in \mathcal{B}_l)\, df.$$

Mean Square Error (MSE): $d_1(f,g) = (f-g)^2$

$$\sigma_q^2 = E\{|\mathcal{F} - Q(\mathcal{F})|^2\} = \sum_{l\in\mathcal{L}} P(\mathcal{B}_l) \int_{b_{l-1}}^{b_l} (f - g_l)^2\, p(f \mid \mathcal{B}_l)\, df.$$

Uniform Quantization



Uniform source:

$$p(f) = \begin{cases} 1/B & f \in (f_{\min}, f_{\max}) \\ 0 & \text{otherwise} \end{cases}$$

$$\sigma_q^2 = \frac{q^2}{12} = \sigma_f^2 \, 2^{-2R}$$

$$\text{SNR} = 10 \log_{10} \frac{\sigma_f^2}{\sigma_q^2}$$

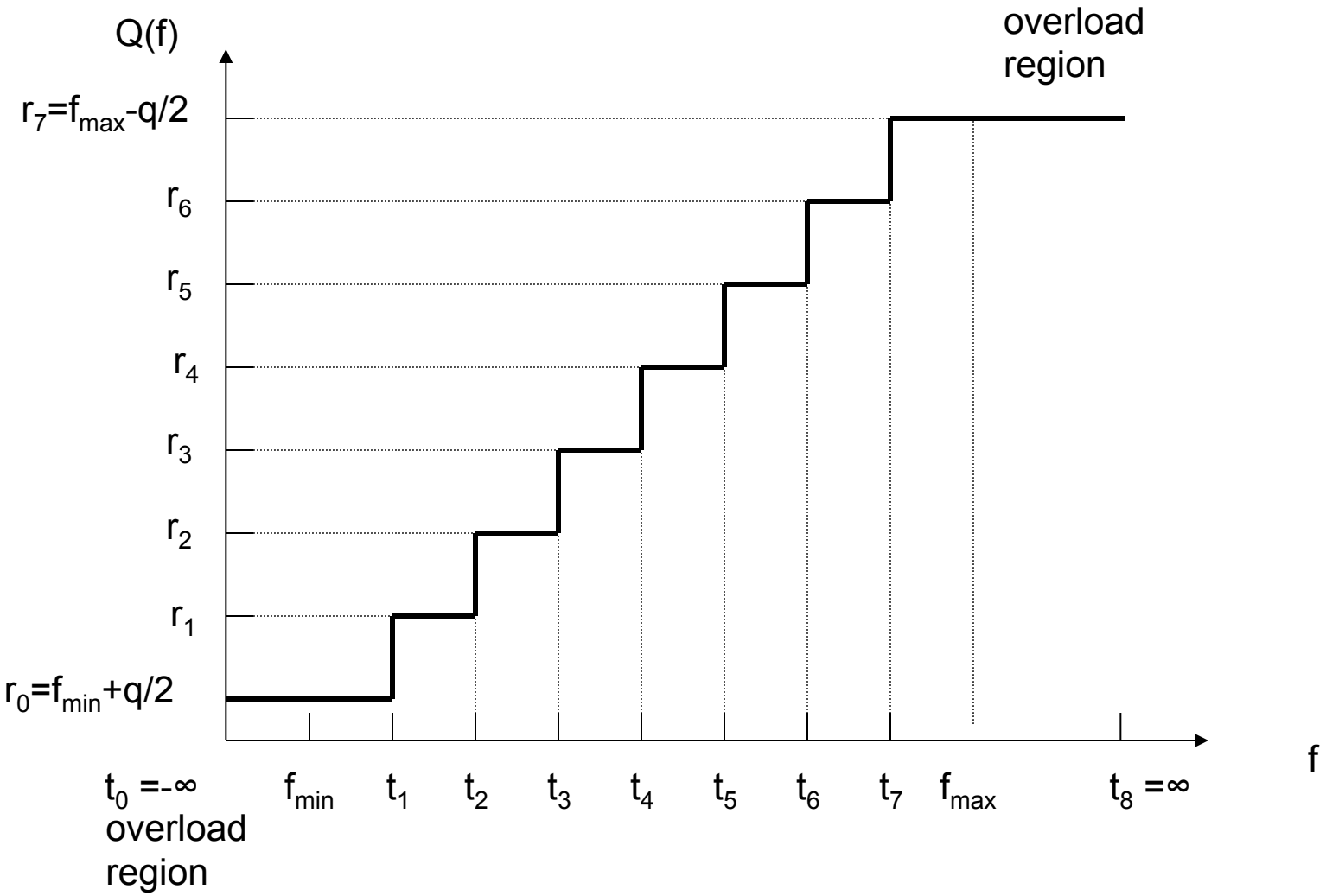
$$= (20 \log_{10} 2) \; R$$

$$= 6.02 R \text{ (dB)}$$

$$Q(f) = \left\lfloor \frac{f - f_{\min}}{q} \right\rfloor * q + \frac{q}{2} + f_{\min},$$

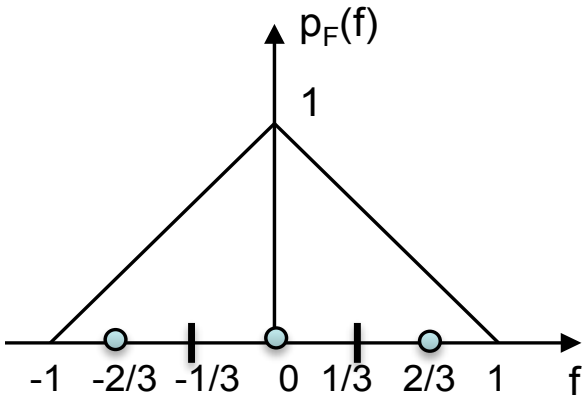
Each additional bit provides 6dB gain!

Truncated uniform quantization for sources with infinite range



Example

- Suppose the signal has the following distribution. We use a uniform quantizer with three levels, as indicated below. What is the quantization MSE?



Minimum MSE (MMSE) Quantizer

Determine b_l, g_l to minimize MSE

$$\sigma_q^2 = E\{|\mathcal{F} - Q(\mathcal{F})|^2\} = \sum_{l \in \mathcal{L}} P(\mathcal{B}_l) \int_{b_{l-1}}^{b_l} (f - g_l)^2 p(f \mid \mathcal{B}_l)\, df.$$

Setting $\frac{\partial \sigma_q^2}{b_l} = 0, \frac{\partial \sigma_q^2}{g_l} = 0$ yields:

$$b_l = \frac{g_l + g_{l+1}}{2}, \quad \text{or} \quad \mathcal{B}_l = \{f : d_1(f, g_l) \le d_1(f, g_l'), \forall l' \neq l\}.$$

(Nearest Neighbor Condition)

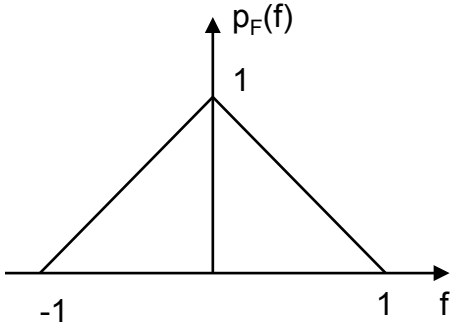
$$g_l = E\{\mathcal{F} \mid \mathcal{F} \in \mathcal{B}_l\} = \int_{\mathcal{B}_l} f\, p(f \mid f \in \mathcal{B}_l)\, df.$$

(Centroid Condition)

- Special case: uniform source
  - MSE optimal quantizer = Uniform quantizer

Example

- Going back to the previous example. What is the MMSE quantizer (partition levels, reconstruction levels) and corresponding MSE?



High Resolution Approximation of MMSE Quantizer

- For a source with arbitrary pdf, when the rate is high so that the pdf within each partition region can be approximated as flat:

$$\sigma_q^2 = \epsilon^2 \sigma_f^2 2^{-2R}$$

$$\epsilon^2 = \frac{1}{12}\left(\int_{-\infty}^{\infty} \tilde{p}(f)^{1/3}\, df\right)^3, \quad \tilde{p}(f) = \sigma_f p(\sigma_f f)$$

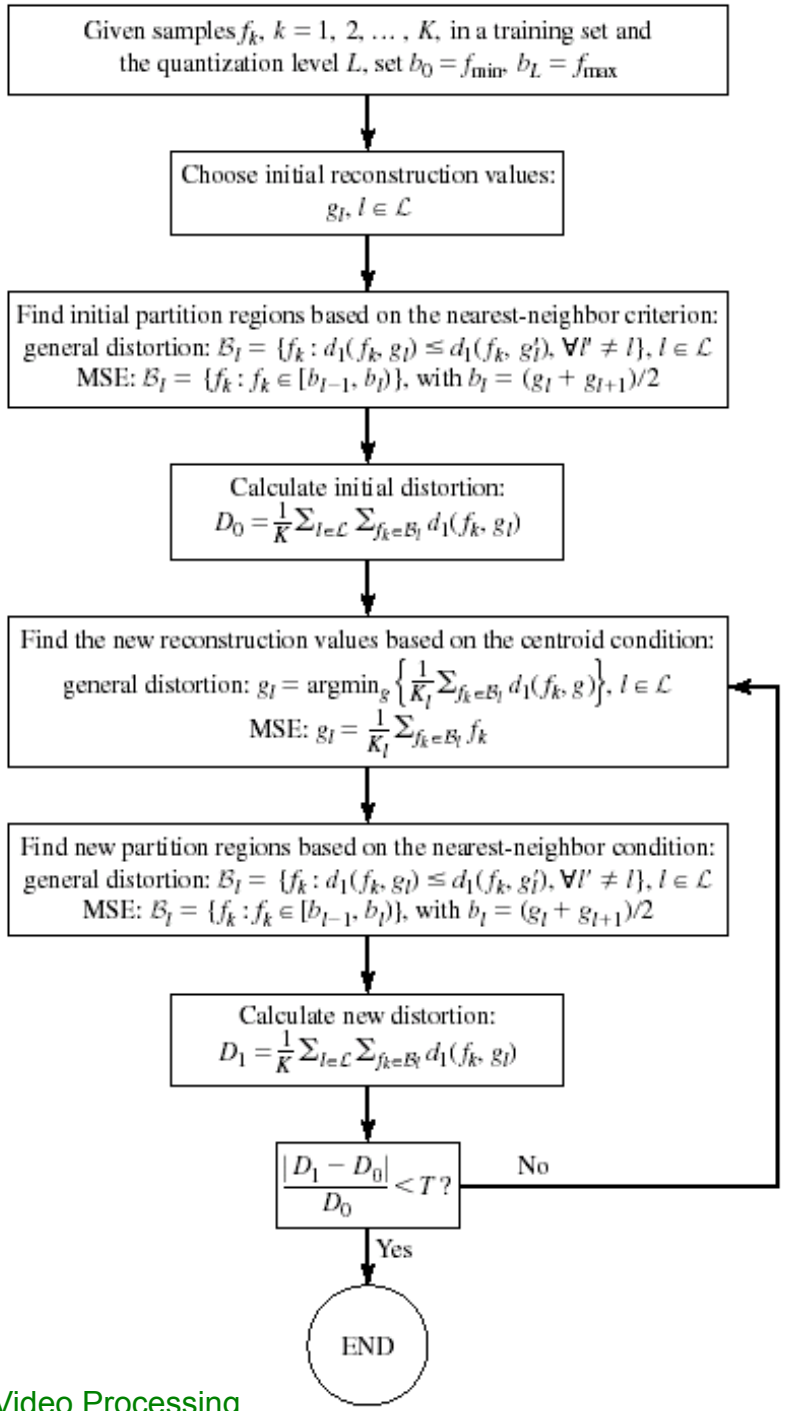
Uniform source: $\varepsilon^2 = 1$

i.i.d Gaussian source: $\varepsilon^2 = 2.71$ (w/o VLC)

Bound for Gaussian source: $\varepsilon^2 = 1$

Lloyd Algorithm

- In general, one may not be able to find closed-form optimal solution given the signal pdf.
- Lloyd algorithm is an iterative algorithms for determining MMSE quantizer parameters
- Can be based on a pdf or training data
- Iterate between centroid condition and nearest neighbor condition



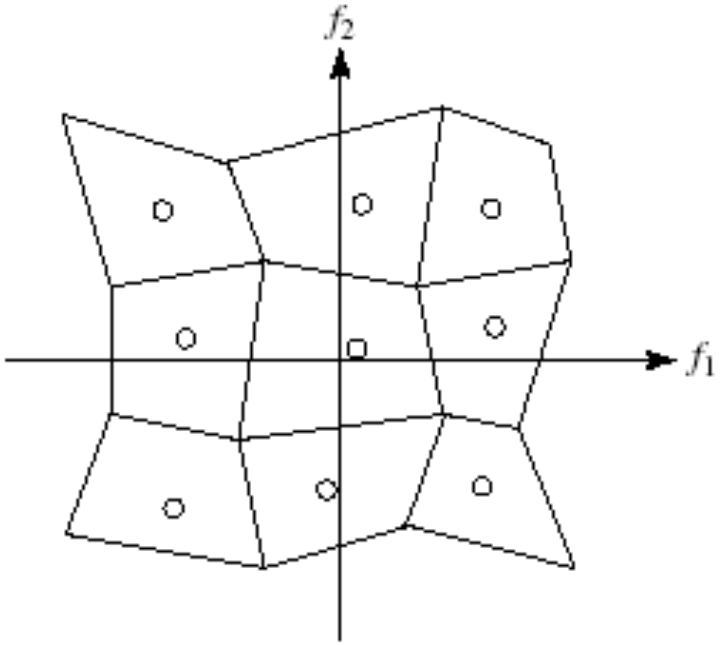
Vector Quantization

- General description
- Nearest neighbor quantizer
- MMSE quantizer
- Generalized Lloyd algorithm

Vector Quantization: General Description

- Motivation: quantize a group of samples (a vector) together, to exploit the correlation between these samples
- Each sample vector is replaced by one of representative vectors (or patterns) that often occur in the signal
- Applications:
  - Color quantization: Quantize all colors appearing in an image to *L* colors for display on a monitor that can only display *L* distinct colors at a time – Adaptive palette
  - Image quantization: Quantize every *NxN* block into one of the *L* typical patterns (obtained through training). More efficient with larger block size, but block size are limited by complexity.

VQ as Space Partition



Original vector: $\mathbf{f} \in R^N$

Quantization levels: L

Partition regions: B_l

Reconstruction vector (codeword): $\mathbf{g}_l$

Quantizer mapping: $Q(\mathbf{f}) = \mathbf{g}_l$, if $\mathbf{f} \in B_l$

Codebook: $C = \{\mathbf{g}_l, l = 1,2,...,L\}$

Bit rate: $R = \frac{1}{N}\log_2 L$

Every point in a region (B_l) is replaced by (quantized to) the point indicated by the circle ($\mathbf{g}_l$)

Distortion Measure

General measure:

$$D_q = E\{d_N(\mathcal{F}, Q(\mathcal{F}))\} = \int_{\mathcal{B}} p_N(\mathbf{f}) d_N(\mathbf{f}, Q(\mathbf{f}))\, d\mathbf{f}$$

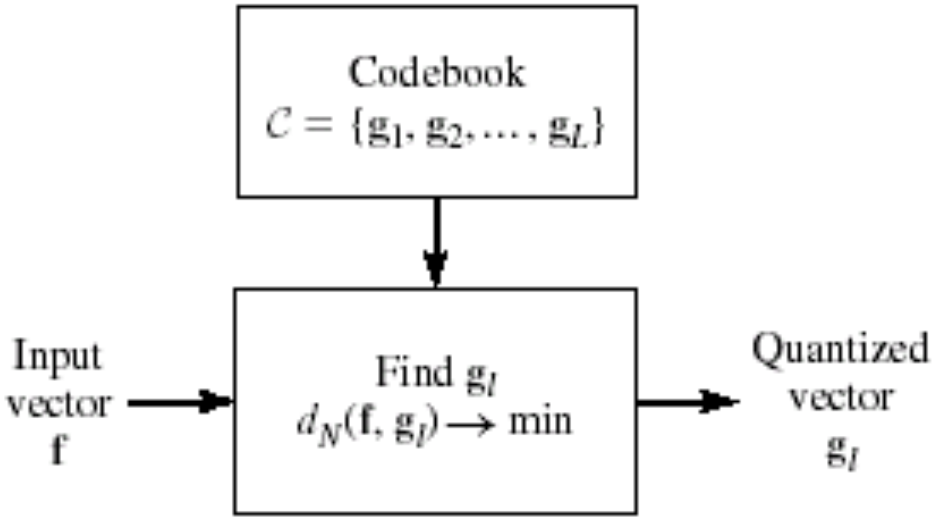
$$= \sum_{l=1}^{L} P(\mathcal{B}_l)\, D_{q,l}$$

$$D_{q,l} = E\{d_N(\mathcal{F}, Q(\mathcal{F})) \mid \mathcal{F} \in \mathcal{B}_l\} = \int_{\mathbf{f}\in\mathcal{B}_l} p_N(\mathbf{f} \mid \mathbf{f} \in \mathcal{B}_l) d_N(\mathbf{f}, \mathbf{g}_l)\, d\mathbf{f}.$$

MSE:

$$d_N(\mathbf{f}, \mathbf{g}) = \frac{1}{N} \sum_{n=1}^{N} (f_n - g_n)^2,$$

Nearest Neighbor (NN) Quantizer



$$\mathcal{B}_l = \{\mathbf{f} \in \mathcal{R}^N : d_N(\mathbf{f}, \mathbf{g}_l) \leq d_N(\mathbf{f}, \mathbf{g}'_l), \forall l' \neq l\}.$$

Challenge: How to determine the codebook?

Complexity of NN VQ

- Complexity analysis:
  - Must compare the input vector with all the codewords
  - Each comparison takes N operations
  - Need L=2^{NR} comparisons
  - Total operation = N 2^{NR}
  - Total storage space = N 2^{NR}
  - Both computation and storage requirement increases exponentially with N!
- Example:
  - N=4x4 pixels, R=1 bpp: 16x2^16=2^20=1 Million operation/vector
  - Apply to video frames, 720x480 pels/frame, 30 fps: 2^20*(720x480/16)*30=6.8 E+11 operations/s !
  - When applied to image, block size is typically limited to <= 4x4
- Fast algorithms:
  - Structured codebook so that one can conduct binary tree search
  - Product VQ: can search subvectors separately

MMSE Vector Quantizer

- Necessary conditions for MMSE
  - Nearest neighbor condition

$$\mathcal{B}_l = \{\mathbf{f} : d_N(\mathbf{f}, \mathbf{g}_l) \le d_N(\mathbf{f}, \mathbf{g}_l'), \forall l' \ne l\}.$$

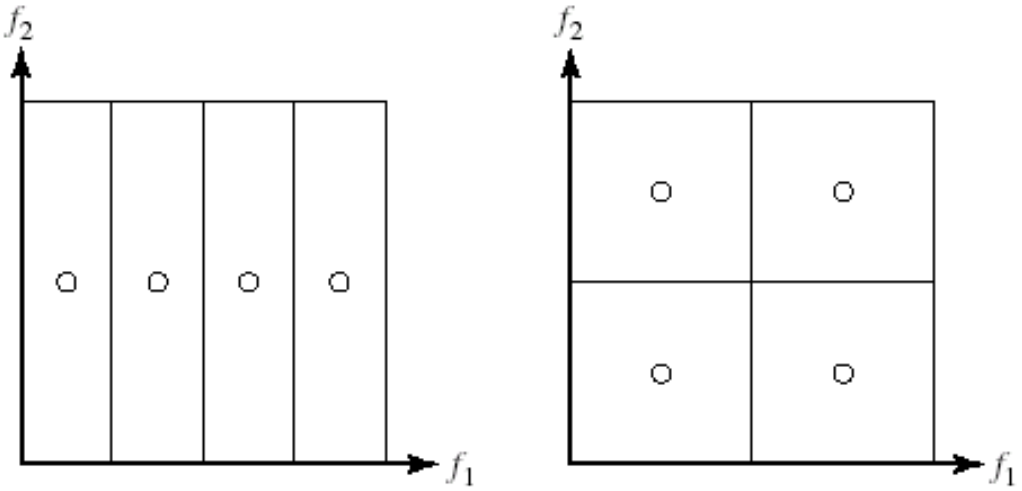
  - Generalized centroid condition:

$$\mathbf{g}_l = \text{argmin}_{\mathbf{g}} E\{d_N(\mathcal{F}, \mathbf{g}) \mid \mathcal{F} \in \mathcal{B}_l\}.$$

  - MSE as distortion:

$$\mathbf{g}_l = \int_{\mathcal{B}_l} p(\mathbf{f} \mid \mathbf{f} \in \mathcal{B}_l)\mathbf{f}\, d\mathbf{f} = E\{\mathcal{F} \mid \mathcal{F} \in \mathcal{B}_l\}.$$

Caveats ☹



Both quantizers satisfy the NN and centroid condition, but the quantizer on the right is better!

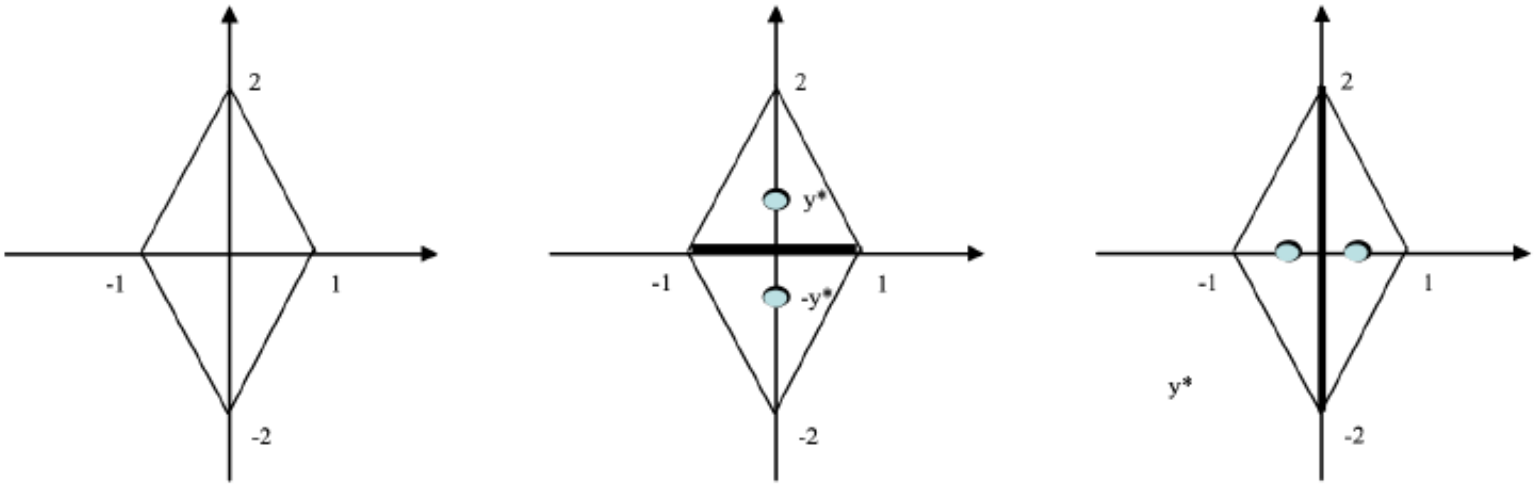
NN and centroid conditions are necessary but NOT sufficient for MSE optimality!

Example

(15 pt) Consider coding a 2-D random vector that is uniformly distributed over the region illustrated in Fig. 1(a). Suppose you want to design a codebook with 2 codewords. One possible codebook construction (codeword locations and region partition) is illustrated in Figure 1(b).

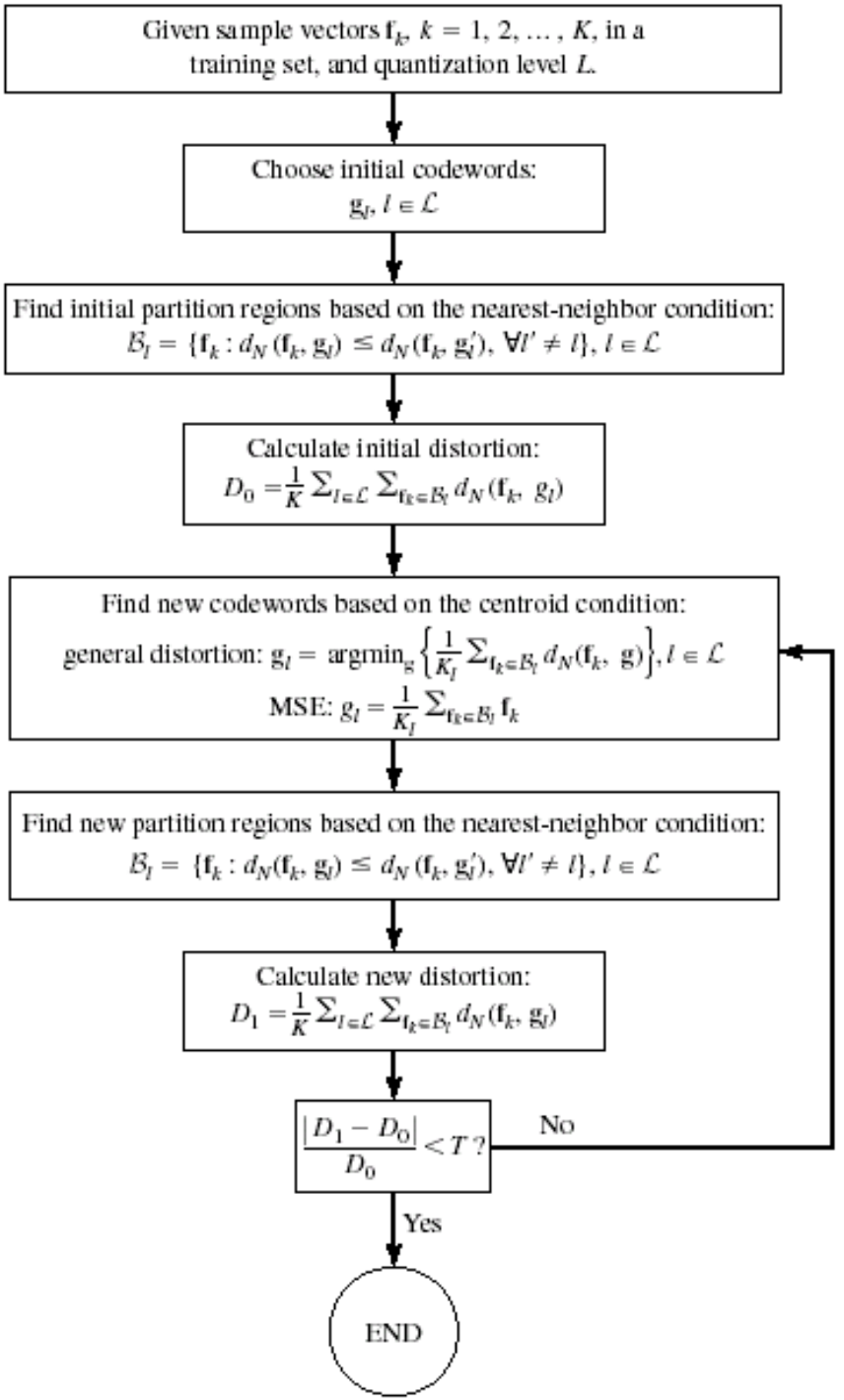
a. Determine the value of y* in the upper triangle in Fig. 1(b) that will minimize the mean square error of the quantizer. Also determine the corresponding minimal mean square error.

b. Another possible codebook configuration is shown in Fig. 1(c). Is this codebook better or worse than that in Fig. 1(b)? why?

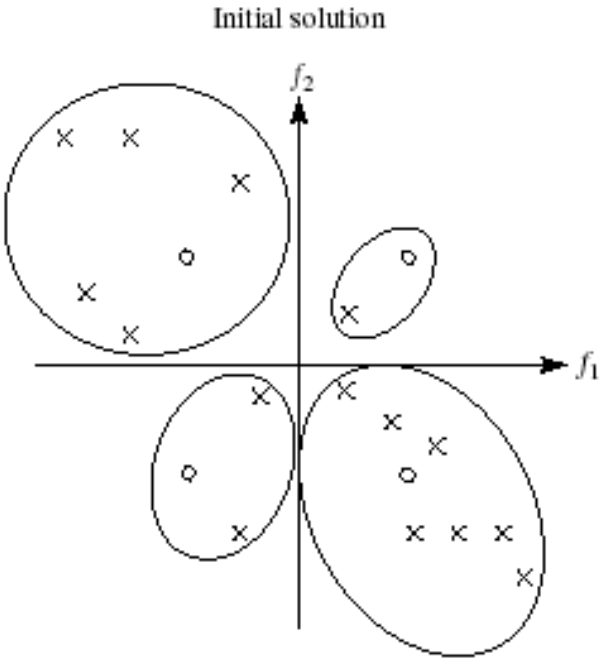


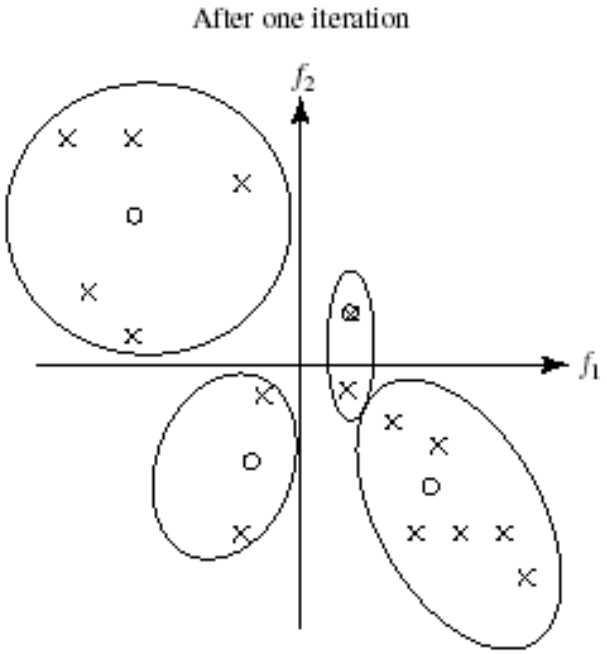
Generalized Lloyd Algorithm (LBG Algorithm)

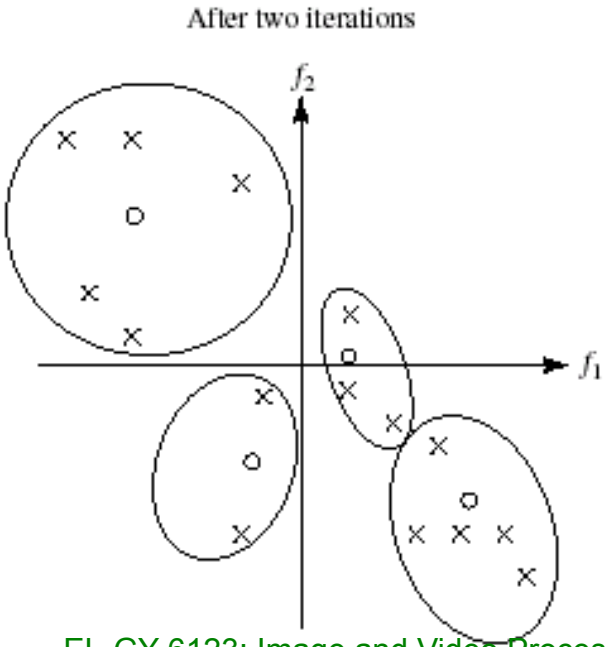
- Start with initial codewords
- Iterate between finding best partition using NN condition, and updating codewords using centroid condition

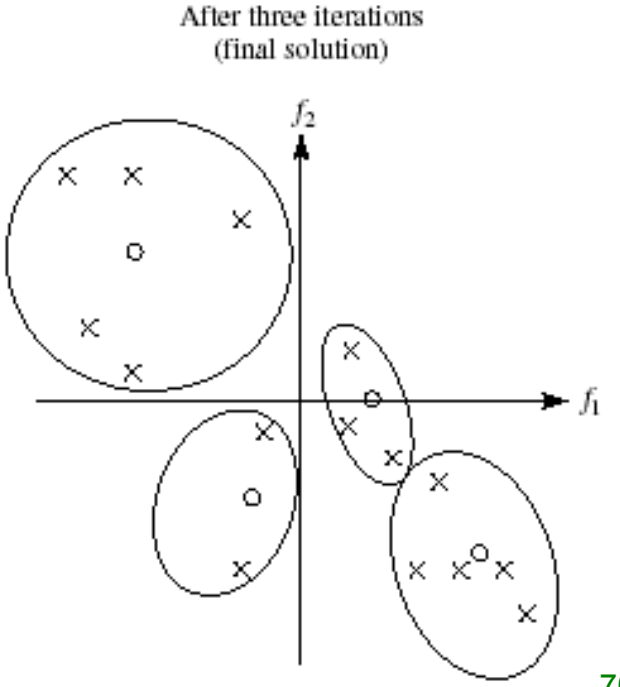


Example


Initial solution
f_2
f_1


After one iteration
f_2
f_1


After two iterations
f_2
f_1


After three iterations
(final solution)
f_2
f_1

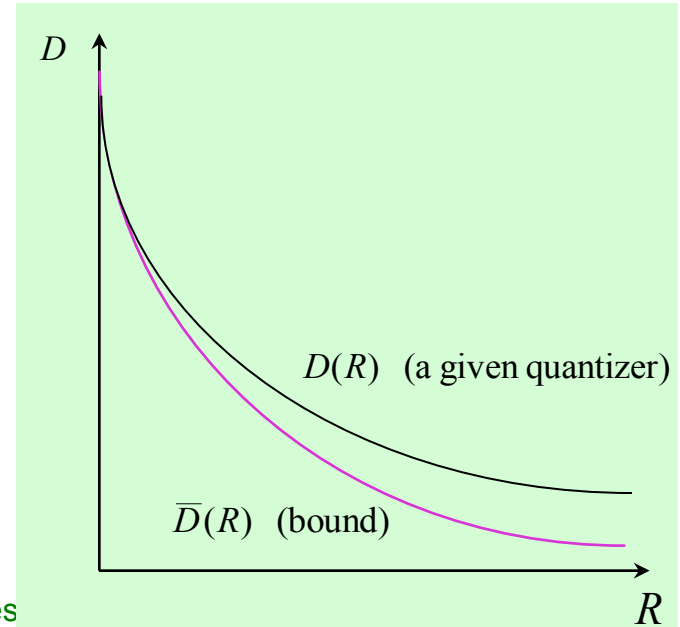
Rate-Distortion Characterization of Lossy Coding

- Operational rate-distortion function of a quantizer:
  - Relates rate and distortion: R(D)
  - A vector quantizer reaches a different point on its R(D) curve by using a different number of codewords
  - Can also use distortion-rate function D(R)
- Rate distortion bound for a source
  - Minimum rate R needed to describe the source with distortion <=D

$$\bar{R}(D) = \lim_{N\to\infty} \min_{q_N(\mathbf{g}|\mathbf{f})\in Q_{D,N}} R_N(D; q_N(\mathbf{g}|\mathbf{f}))$$

$$Q_{D,N} = \{q_N(\mathbf{g}|\mathbf{f}) : E\{d_N(\mathcal{F}, \mathcal{G})\} \le D\}$$

- RD optimal quantizer:
  - Minimize D for given R or vice versa



Lossy Coding Bound
(Shannon Lossy Coding Theorem, Not required)

$$\bar{R}(D) = \lim_{N\to\infty} \min_{q_N(\mathbf{g}|\mathbf{f})\in Q_{D,N}} R_N(D; q_N(\mathbf{g}|\mathbf{f}))$$

$$Q_{D,N} = \{q_N(\mathbf{g}|\mathbf{f}) : E\{d_N(\mathcal{F}, \mathcal{G})\} \le D\}$$

$$\bar{R}(D) = \lim_{N\to\infty} \min_{q_N(\mathbf{g}|\mathbf{f})\in Q_{D,N}} \frac{1}{N} I_N(\mathcal{F}; \mathcal{G}).$$

$I_N(\boldsymbol{F},\boldsymbol{G})$: mutual information between F and G, information provided by G about F

$Q_{D,N}$: all coding schemes (or mappings q(**g**|**f**)) that satisfy distortion criterion $d_N(f,\boldsymbol{g}) <= D$

$$\bar{R}_L(D) \le \bar{R}(D) \le \bar{R}_G(D),$$

$$\bar{R}_L(D) = \bar{h}(\mathcal{F}) - \frac{1}{2}\log_2 2\pi e D = \frac{1}{2}\log_2 \frac{Q(\mathcal{F})}{D},$$

$h(\boldsymbol{F})$: differential entropy of source $\boldsymbol{F}$

$R_G(D)$: RD bound for Gaussian source with the same variance

i.i.d. Gaussian source requires highest bit rate!

RD Bound for Gaussian Source (Not required)

- i.i.d. 1-D Gaussian:

$$\bar{D}(R) = \sigma^2 2^{-2R}.$$

- i.i.d. N-D Gaussian with independent components:

$$\bar{D}(R) = \left(\prod_n \sigma_n^2\right)^{1/N} 2^{-2R}.$$

- N-D Gaussian with covariance matrix **C**:

$$\bar{D}(R) = \left(\prod_n \lambda_n\right)^{1/N} 2^{-2R} = |\det[\mathbf{C}]|^{1/N} 2^{-2R}.$$

- Gaussian source with power spectrum (FT of correlation function) $S(e^{j\omega})$

$$\bar{R}(D) = \frac{1}{4\pi}\int_{-\pi}^{\pi} \log_2 \frac{S(e^{j\omega})}{D}\, d\omega.$$

Summary on Quantization

- Scalar quantization:
  - Uniform quantizer
  - MMSE quantizer (Nearest neighbor and centroid condition)
    - Closed-form solution for some pdf
    - Lloyd algorithm for numerical solution
- Vector quantization
  - Nearest neighbor quantizer
  - MMSE quantizer (Nearest neighbor and centroid condition)
  - Generalized Lloyd alogorithm
  - Uniform quantizer
    - Can be realized by lattice quantizer (not discussed here)
- Rate distortion characterization of lossy coding (not required)
  - Bound on lossy coding
  - Operational RD function of practical quantizers

References

- Reading assignment:
  - [Wang2002] Sec. 8.1-8.4 (Sec. 8.3.2,8.3.3 optional)
  - [Wang2002] Sec. 8.5-8.7
  - Optional: [Woods2012] Sec. 9.3, 9.4, Appendix on Information Theory
- Optional reading on arithmetic coding and CABAC
  - Witten, Radford, Neal, Cleary, “Arithmetic Coding for Data Compression” Communications of the ACM, vol. 30, no. 6, pp. 520-540, June 1987.
  - Marpe, Detlev, Heiko Schwarz, and Thomas Wiegand. "Context-based adaptive binary arithmetic coding in the H. 264/AVC video compression standard." Circuits and Systems for Video Technology, IEEE Transactions on 13.7 (2003): 620-636.
  - http://www.hhi.fraunhofer.de/fields-of-competence/image-processing/research-groups/image-video-coding/statistical-modeling-coding/fast-adaptive-binary-arithmetic-coding-m-coder.html

Written Assignment (1)

- Problems from [Wang2002] Prob. 8.1,8.6, 8.11, 8.14
- Additional problems in the following slides

Written assignment (2)

(15 pt) Consider coding a 2-D random vector that is distributed over a triangular region illustrated in Fig. (a) with the following distribution function: $p(x,y) = A\exp(-|x|-|y|)$ where A is some constant to normalize the distribution function. Suppose you want to design a codebook with 2 codewords. Figures (b) and (c) illustrate two possible codebooks with their corresponding region partitions.

a. Which codebook will likely yield less quantization error? Why?
b. Does either codebook and associated partition satisfy the nearest neighbor condition?
c. For the codebook in Fig. (b), determine the x- and y- coordinate of each codeword (indicated by a and b, respectively) that will minimize the mean square error. Determine the minimal mean square error. If you run out of time, you can leave your solution in terms of some integrals, without getting the explicit solution. You should make use of the symmetry to simplify your solution.

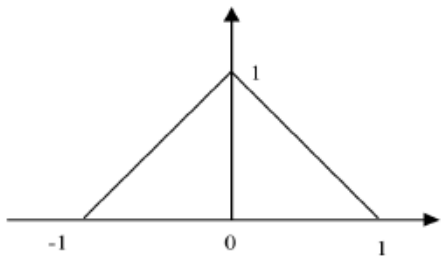


Fig. (a)

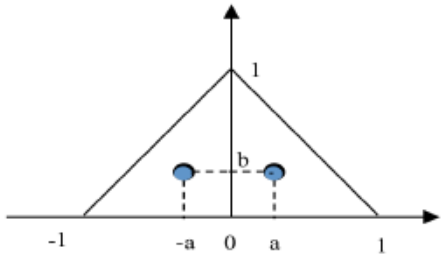


Fig. (b)

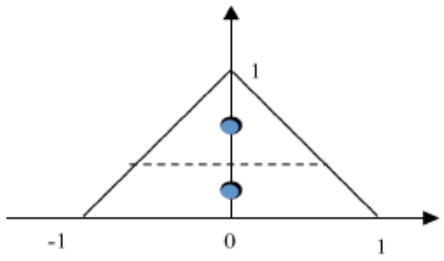


Fig. (c)

Written assignment (3)

(15 pt) Vector Quantization

a. (5 pt) What is the operation count of a nearest neighbor vector quantizer with vector dimension N and bit rate R bits/sample? Consider one addition, one subtraction, one multiplication each as one operation.

b. (10 pt) In order to reduce the complexity, we can code the norm (or gain) of an input vector using a scalar quantizer and the normalized vector (or shape vector) using a vector quantizer. This method is called *gain-shape vector quantizer*. Suppose an input vector is $\mathbf{f} = [f_1, f_2, \ldots, f_N]$, its norm (called gain factor) is defined as $G = \left(\sum_n f_n^2 \right)^{1/2}$, the normalized vector is $\tilde{\mathbf{f}} = \frac{1}{G}\mathbf{f}$. If we use R_1 bits to quantize the gain factor G, and R_2 bits/sample to quantize the normalized vector $\tilde{\mathbf{f}}$, where R_1 and R_2 are chosen such that the total bit rate (bits/sample) is the same as in (a) (that is, $R_1 + NR_2 = NR$), what will be the total operation count for this method? What is the saving factor compared to direct vector quantization? (Consider square root as one operation, also assume the scalar quantizer is in general non-uniform, and you need to use the nearest neighbor rule to determine the quantized level.)

Computer assignment (Optional!)

- Do one of the two
  - Option 1: Write a program to perform vector quantization on a gray scale image using 4x4 pixels as a vector. You should design your codebook using all the blocks in the image as training data, using the generalized Lloyd algorithm. Then quantize the image using your codebook. You can choose the codebook size, say, L=128 or 256. If your program can work with any specified codebook size L, then you can observe the quality of quantized images with different L.
  - Option 2: Write a program to perform color quantization on a color RGB image. Your vector dimension is now 3, containing R,G,B values. The training data are the colors of all the pixels. You should design a color palette (i.e. codebook) of size L, using generalized Lloyd algorithm, and then replace the color of each pixel by one of the color in the palette. You can choose a fixed L or let L be a user-selectable variable. In the later case, observe the quality of quantized images with different L.