Image/Video Compression: Introduction & Review

Trac D. Tran
ECE Department
The Johns Hopkins University
Baltimore, MD 21218

Course Overview

- Image/Video Compression and Communications
  - Fundamentals: motivation, signal properties & formats, information theory, variable-length coding, quantization
  - Transform coding framework, JPEG, JPEG2000
  - Video coding and international video standards
  - Image/Video communications
- Goals
  - Focus on big pictures, key concepts, elegant ideas, no rigorous treatment
  - Provide hands-on experience with simple Matlab exercises
  - Illustrate applications of the previous two DSPSS courses!
  - Hopefully lead to future research and developments

Outline

- Image/video properties & formats
  - Image/video signals: properties & formats, color spaces
  - General image/video coding framework
  - Error & similarity measurements
  - Statistical modeling of image/video signals
- Lossless variable-length coding
  - Information theory, entropy, entropy coding
  - Huffman coding
  - Arithmetic coding
- Lossy coding
  - Quantization: optimal conditions, scalar quantization, embedded quantization

Image/Video Everywhere!

- Fax machines: transmission of binary images
- Digital cameras: still images
- Digital camcorders: video sequences with audio
- Digital television broadcasting
- Digital video disk (DVD)
- Personal video recorder (PVR, TiVo)
- Images on the World Wide Web
- Video streaming
- Video conferencing
- Image/video on cell phones, PDAs
- High-definition televisions (HDTV)
- Medical imaging: X-ray, MRI, ultrasound, telemedicine
- Military imaging: multi-spectral, satellite, infrared, microwave

Digital Bit Rates

- *A picture is worth a thousand words?*
- Size of a typical color image
  - For display
    - 640 x 480 x 24 bits = 7372800 bits = 92160 bytes
  - For current mainstream digital cameras (5 Mega-pixel)
    - 2560 x 1920 x 24 bits = 117964800 bits = 14745600 bytes
  - For an average word
    - 4-5 characters/word, 7 bits/character: 32 bits ~= 4 bytes
- Bit rate: bits per second for transmission
  - Raw digital video (DVD format)
    - 720 x 480 x 24 x 24 frames: ~200 Mbps
  - CD Music
    - 44100 samples/second x 16 bits/sample x 2 channels ~ 1.4 Mbps

Reasons for Compression

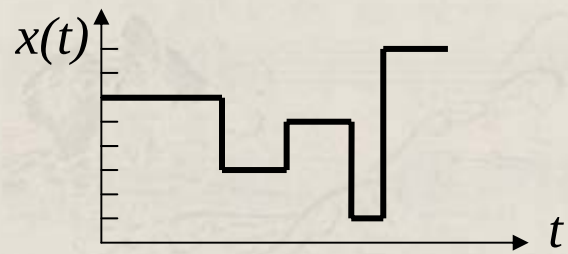
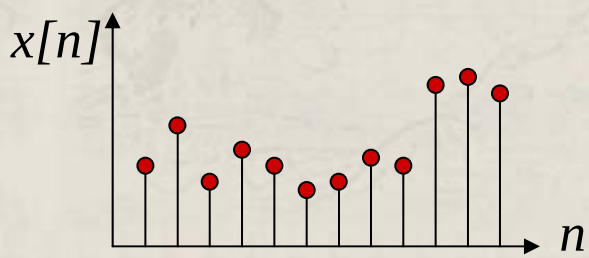
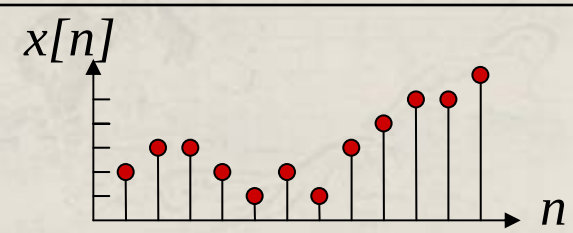
- Digital bit rates
  - Terrestrial TV broadcasting channel: ~20 Mbps
  - DVD: 10...20 Mbps
  - Ethernet/Fast Ethernet: <10/100 Mbps
  - Cable modem downlink: 1-3 Mbps
  - DSL downlink: 384...2048 kbps
  - Dial-up modem: 56 kbps max
  - Wireless cellular data: 9.6...384 kbps
- Compression = Efficient data representation!
  - Data need to be accessed at a different time or location
  - Limited storage space and transmission bandwidth
  - Improve communication capability

Personal Video Recorder (PVR)

MPEG2 Quality

Best	7.7 Mbps
High	5.4 Mbps
Medium	3.6 Mbps
Basic	2.2 Mbps

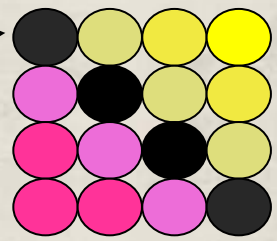
Continuous & Discrete Representations

	Continuous-Amplitude	Discrete-Amplitude
Continuous -Time (Space)	Local telephone, cassette-tape recording & playback, phonograph, photograph	 telegraph
Discrete -Time (Space)	 Switched capacitor filter, speech storage chip, half-tone photography	 CD, DVD, cellular phones, digital camera & camcorder, digital television, inkjet printer

Multi-Dimensional Digital Signals

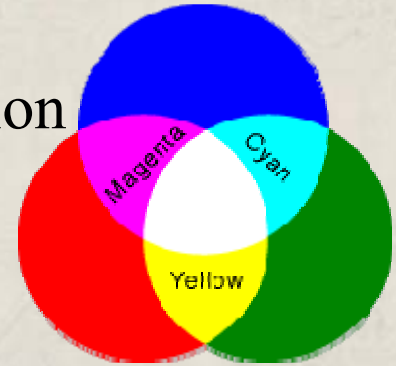
- **Images**: 2-D digital signals

pixel
or
pel

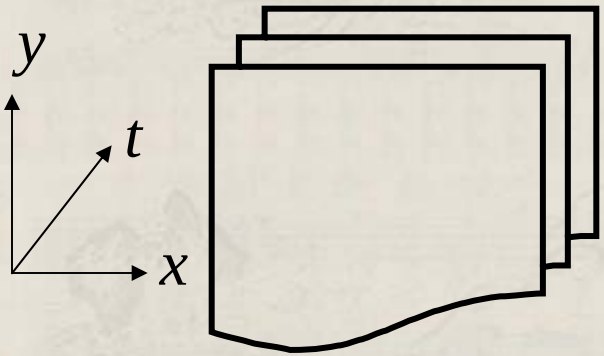


black p=0 | gray p=128 | white p=255

colors: combination of RGB



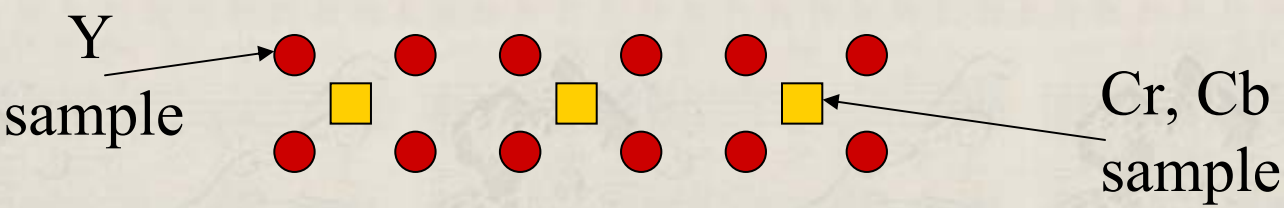
- **Video Sequences**: 3-D digital signals, a collection of 2-D images called frames



Color Spaces: RGB & YCrCb

- RGB
  - Red Green Blue, typically 8-bit per sample for each color plane
- YCrCb
  - Y: luminance, gray-scale component
  - Cr & Cb: chrominance, color components, less energy than Y
  - Chrominance components can be down-sampled without much aliasing
  - YCrCb, also known as YPrPb, is used in component video

$$\begin{bmatrix} Y \\ C_R \\ C_B \end{bmatrix} = \begin{bmatrix} 0.257 & 0.504 & 0.098 \\ 0.439 & -0.368 & -0.071 \\ -0.148 & -0.291 & 0.439 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix}$$



Another Color Space: YUV

- YUV is another popular color space, similarly to YCrCb
  - Y: luminance component
  - UV: color components
  - YUV is used in PAL/NTSC broadcasting

$$\begin{bmatrix} Y \\ C_R \\ C_B \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

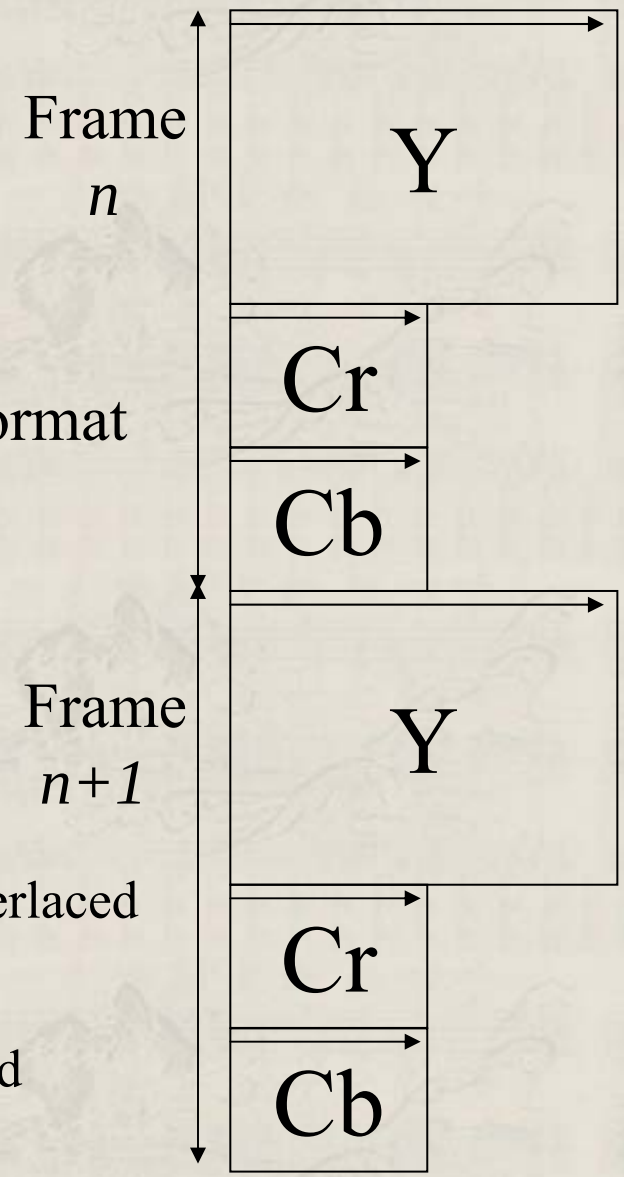
Y: 176 x 144

U: 88 x 72

V: 88 x 72

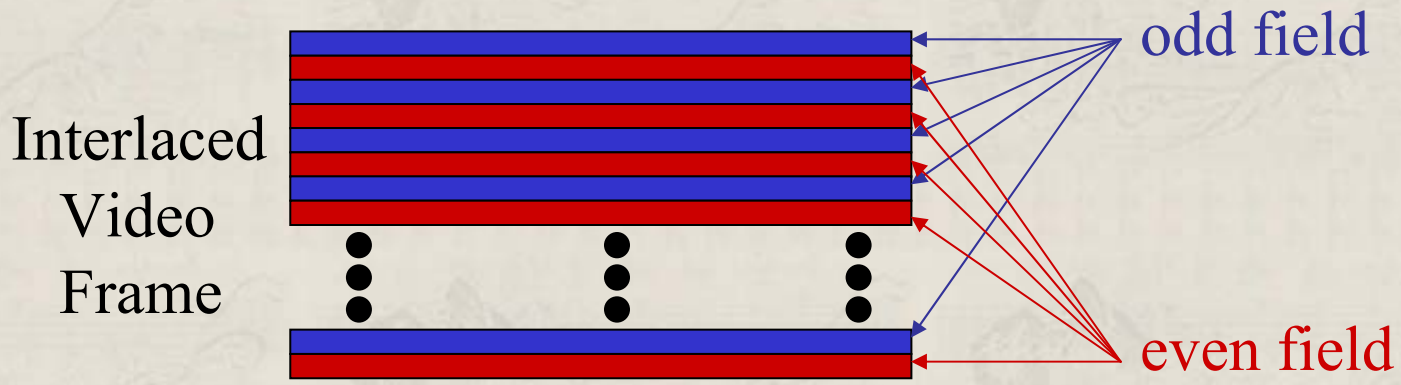
Popular Signal Formats

- CIF: Common Intermediate Format
  - Y resolution: 352 x 288
  - CrCb/UV resolution: 176 x 144
  - Frame rate: 30 frames/second progressive
  - 8 bits/pixel(sample)
- QCIF: Quarter Common Intermediate Format
  - Y resolution: 176 x 144
  - CrCb/UV resolution: 88 x 72
  - Frame rate: 30 frames/second progressive
  - 8 bits/pixel (sample)
- TV – NTSC
  - Resolution: 704 x 480, 30 frames/second interlaced
- DVD – NTSC
  - Resolution: 720 x 480, 24 – 30 frames/second progressive



High-Definition Television (HDTV)

- 720i
  - Resolution: 1280 x 720, interlaced
- 720p
  - Resolution: 1280 x 720, progressive
- 1080i
  - Resolution: 1920 x 1080, interlaced
- 1080p
  - Resolution: 1920 x 1080, progressive



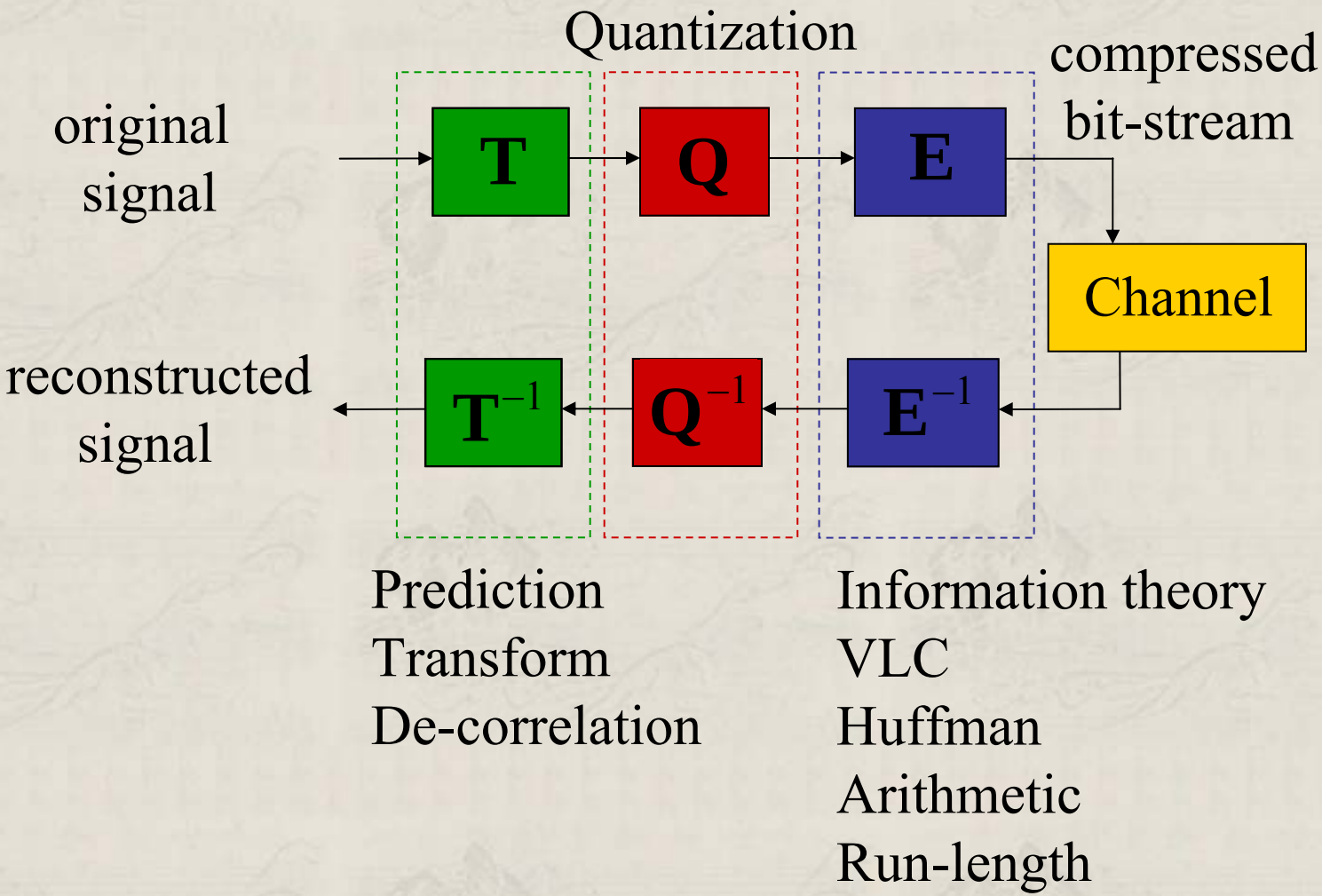
Examples of Still Images

Examples of Video Sequences

Frame 1 51 71 91 111

- Observations of Visual Data
  - There is a lot of redundancy, correlation, strong structure within natural image/video
  - Images
    - Spatial correlation: a lot of smooth areas with occasional edges
  - Video
    - Temporal correlation: neighboring frames seem to be very similar

Image/Video Compression Framework

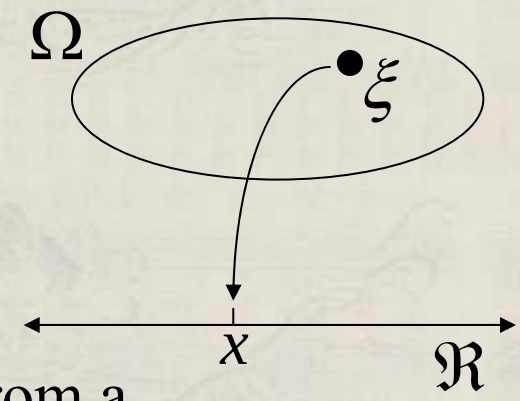


Deterministic versus Random

- Deterministic
  - Signals whose values can be specified explicitly
  - Example: a sinusoid
- Random
  - Digital signals in practice can be treated as a collection of random variables or a random process
  - The symbols which occur randomly carry information
- Probability theory
  - The study of random outcomes/events
  - Use mathematics to capture behavior of random outcomes and events

Random Variable

- Random variable (RV)
  - A random variable X is a mapping which assigns a real number x to each possible outcome of a random experiment ξ
  - A random variable X takes on a value x from a given set. Thus it is simply an event whose outcomes have numerical values
  - Examples
    - X in coin toss, $X=1$ for Head, $X=0$ for Tail
    - The temperature outside our lecture hall at any moment t
    - The pixel value at location x, y in frame n of a future Hollywood blockbuster



Probability Density Function

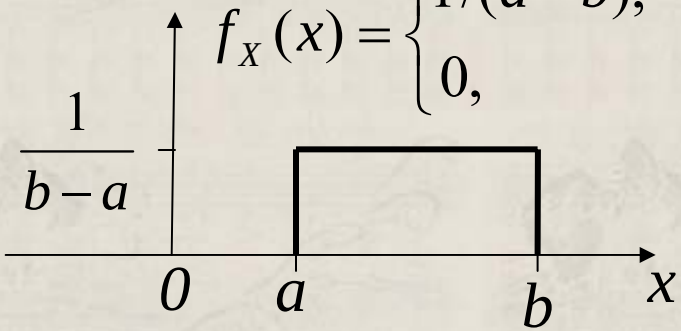
- Probability density function (PDF) of a RV X
  - Function $f_X(x)$ defined such that:

$$P[x_1 \le X \le x_2] = \int_{x_1}^{x_2} f_X(x)dx$$

  - Histogram of X !!!
  - Main properties:
    - $\int_{-\infty}^{\infty} f_X(x)dx = 1$
    - $f_X(x) \ge 0, \forall x$

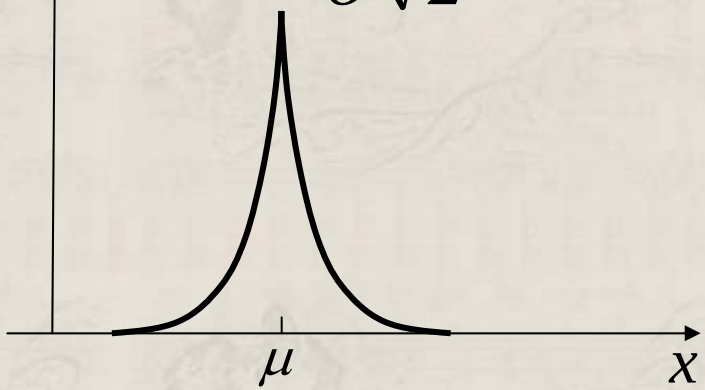
PDF Examples

$$f_X(x) = \begin{cases} 1/(a-b), & a \le x \le b \\ 0, & otherwise \end{cases}$$



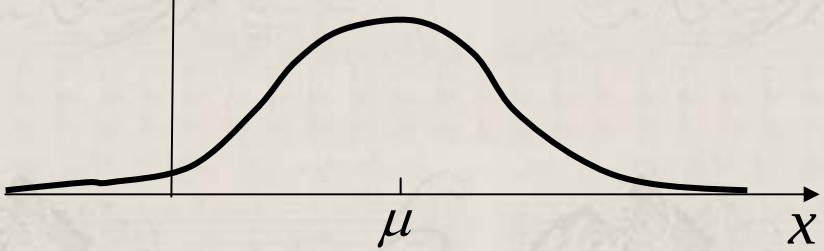
Uniform PDF

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



Laplacian PDF

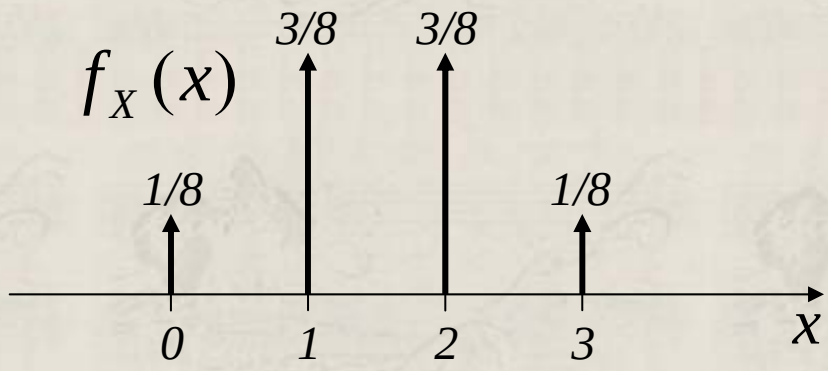
$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$$



Gaussian PDF

Discrete Random Variable

- RV that takes on discrete values only
- PDF of discrete RV = discrete histogram
- Example: how many Heads in 3 independent coin tosses?



$$f_X(x) = \sum_k P_X(x_k)\delta[x - x_k] \qquad \text{with} \qquad P_X(x_k) = P[X = x_k]$$

Expectation

- Expected value
  - Let *g(X)* be a function of RV *X*. The expected value of *g(X)* is defined as

$$E[g(X)] = \int_{-\infty}^{\infty} g(x) f_X(x) dx$$

  - Expectation is linear!
  - Expectation of a deterministic constant is itself: $E[C] = C$
- Mean $\mu_X = E[X] = \int_{-\infty}^{\infty} x f_X(x) dx$
- Mean-square value $E[X^2]$
- Variance $\sigma_X^2 = E[(X - \mu_X)^2]$

$$E[X^2] = \sigma_X^2 + \mu_X^2$$

Cross Correlation & Covariance

- Cross correlation
  - X, Y: 2 jointly distributed RVs
  - Joint PDF:

$$P\left[x_1 \le X \le x_2, y_1 \le Y \le y_2\right] = \int_{y_1}^{y_2}\int_{x_1}^{x_2} f_{XY}(x,y)dxdy$$

  - Expectation:

$$E\left[g(X,Y)\right] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x,y) f_{XY}(x,y)dxdy$$

  - Cross-correlation:

$$R_{XY} \equiv E\left[XY\right]$$

- Cross covariance

$$C_{XY} \equiv E\left[\left(X-\mu_X\right)\left(Y-\mu_Y\right)\right]$$

$$\Rightarrow R_{XY} = C_{XY} + \mu_X \mu_Y$$

Independence & Correlation

- Marginal PDF: $f_X(x) = \int_{-\infty}^{\infty} f_{XY}(x, y)dy$

  $f_Y(y) = \int_{-\infty}^{\infty} f_{XY}(x, y)dx$

- Statistically independent: $f_{XY}(x, y) = f_X(x) f_Y(y)$

- Uncorrelated: $E[XY] = E[X]E[Y], \quad i.e. C_{XY} = 0$

  with 0-mean RVs

- Orthogonal: $E[XY] = 0$

Random Process

- Random process (RP)
  - A collection of RVs
  - A time-dependent RV
  - Denoted *{X[n]}*, *{X(t)}* or simply X[n], X(t)
  - We need N-dimensional joint PDF to characterize *X[n]!*
  - Note: the RVs made up a RP may be dependent or correlated
  - Examples:
    - Temperature *X(t)* outside campus
    - A sequence of binary numbers transmitted over a communication channel
    - Speech, music, image, video signals

Wide-Sense Stationary

- Wide-sense stationary (WSS) random process (RP)
  - A WSS RP is one for which *E[X[n]]* is independent of *n* and $R(m,n) \equiv E[X[m]X[n]]$ only depends on the difference *(m – n)*
  - Mean: $m_X = E[X[n]]$
  - Auto-correlation sequence: $R_{XX}(k) = E[X[n]X[n-k]]$
  - Energy: $E[X^2[n]] = R_{XX}(0)$
  - Variance: $\sigma_X^2 = E\left[(X[n]-m_X)^2\right] \Rightarrow \sigma_X^2 = R_{XX}(0) - m_X^2$
  - Co-variance: $C_{XX}(k) = E[(X[n]-m_X)(X[n-k]-m_X)]$

What happens if the WSS RP has 0-mean?

White Random Process

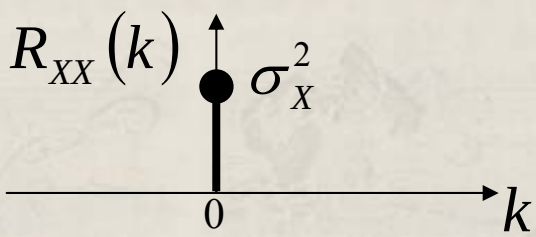
- Power spectral density
  - The power spectrum of a WSS RP is defined as the Fourier transform of its auto-correlation sequence

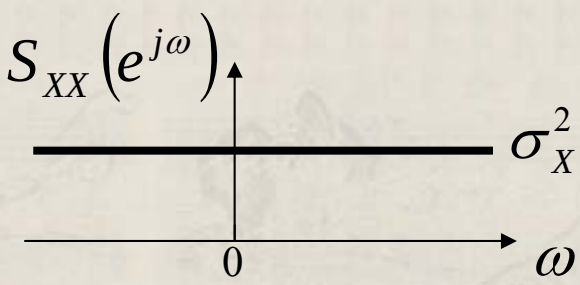
$$S_{XX}\left(e^{j\omega}\right)=\sum_{k}R_{XX}(k)e^{-j\omega k}$$

- White RP
  - A RP is said to be white if any pair of samples are uncorrelated, i.e., $E[X[n]X[m]]=E[X[n]]E[X[m]], m\neq n$
- White WSS RP

$$R_{XX}(k)=\begin{cases} m_X^2, & k\neq 0 \\ \sigma_X^2+m_X^2, & k=0 \end{cases}$$

- White 0-mean WSS RP





Stochastic Signal Model

$w[n]$ → $H(z) = \dfrac{1}{1 - \sum_{n=1}^{N} a_n z^{-n}}$ → $x[n]$

white 0-mean WSS Gaussian noise

AR(N) signal

- For speech: N = 10 to 20
- For images: N = 1! and $a_1 = \rho = 0.95$

$$X(z) = H(z)W(z) = \frac{W(z)}{1 - \rho z^{-1}} \Leftrightarrow X(z) = W(z) + \rho z^{-1} X(z)$$

$$\Leftrightarrow X(z) - \rho z^{-1} X(z) = W(z)$$

$$\Leftrightarrow x[n] = w[n] + \rho x[n-1]$$

AR(1) Signal

Error or Similarity Measures

- Mean Square Error (MSE)

$$\mathrm{L}_2\text{ - norm error}: MSE = \frac{1}{N}\sum_{i=0}^{N-1} E\left(\left|X_i - \hat{X}_i\right|^2\right)$$

- Mean Absolute Difference (MAD)

$$\mathrm{L}_1\text{ - norm error}: MAD = \frac{1}{N}\sum_{i=0}^{N-1} E\left(\left|X_i - \hat{X}_i\right|\right)$$

- Max Error

$$\mathrm{L}_\infty\text{ - norm error}: MaxError = \max_i \left\{E\left(\left|X_i - \hat{X}_i\right|\right)\right\}$$

- Peak Signal-to-Noise Ratio (PSNR)

$$\mathrm{PSNR} = 10\log_{10}\frac{M^2}{MSE};$$

$$M = \text{maximum peak - to - peak value}$$

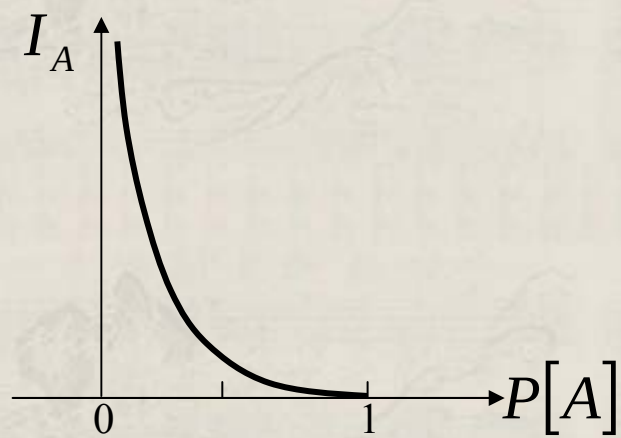
Variable Length Coding: Introduction to Lossless Compression

Trac D. Tran
ECE Department
The Johns Hopkins University
Baltimore, MD 21218

Information Theory

- A measure of information
  - The amount of information in a signal might not equal to the amount of data it produces
  - The amount of information about an event is closely related to its probability of occurrence
- Self-information
  - The information conveyed by an event *A* with probability of occurrence *P[A]* is

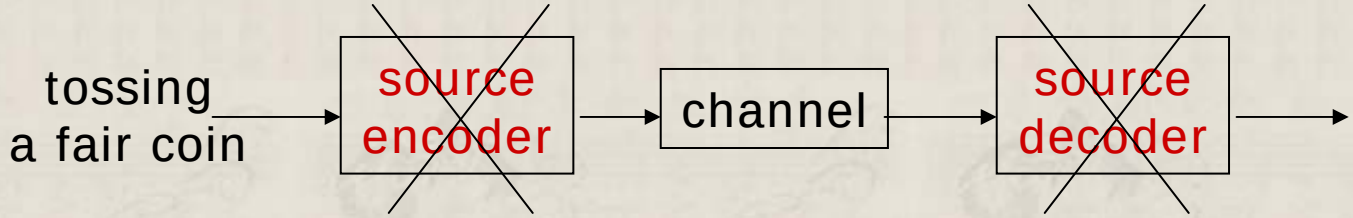
$$I_A = \log_2 \frac{1}{P[A]} = -\log_2 P[A]$$



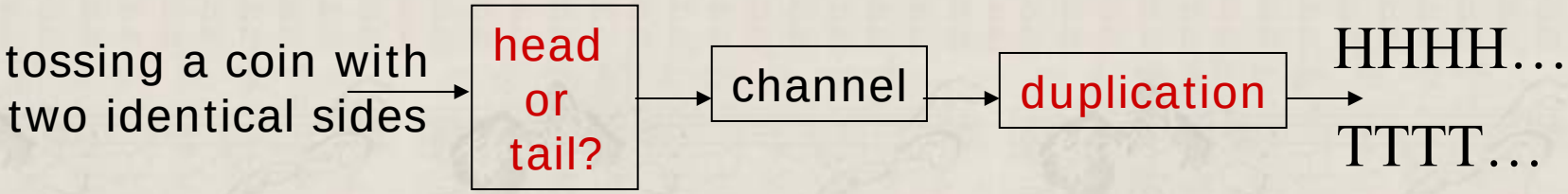
Information = Degree of Uncertainty

- Zero information
  - The sun rises in the east
  - If an integer n is greater than two, then $a^n + b^n = c^n$ has no solutions in non-zero integers a, b, and c
- Little information
  - It will rain in Hà Nội next week
  - Most cellular phones in the future will have multimedia capability
- A lot of information
  - A Vietnamese mathematician proves $P = NP$
  - Việt Nam will win the 2010 World Cup with Quyến Béo winning the Golden Boot!

Two Extreme Cases



$P(X=H)=P(X=T)=1/2$: (maximum uncertainty)
Minimum (zero) redundancy, compression impossible



$P(X=H)=1, P(X=T)=0$: (minimum redundancy)
Maximum redundancy, compression trivial (1 bit is enough)

Redundancy is the opposite of uncertainty

Weighted Self-information

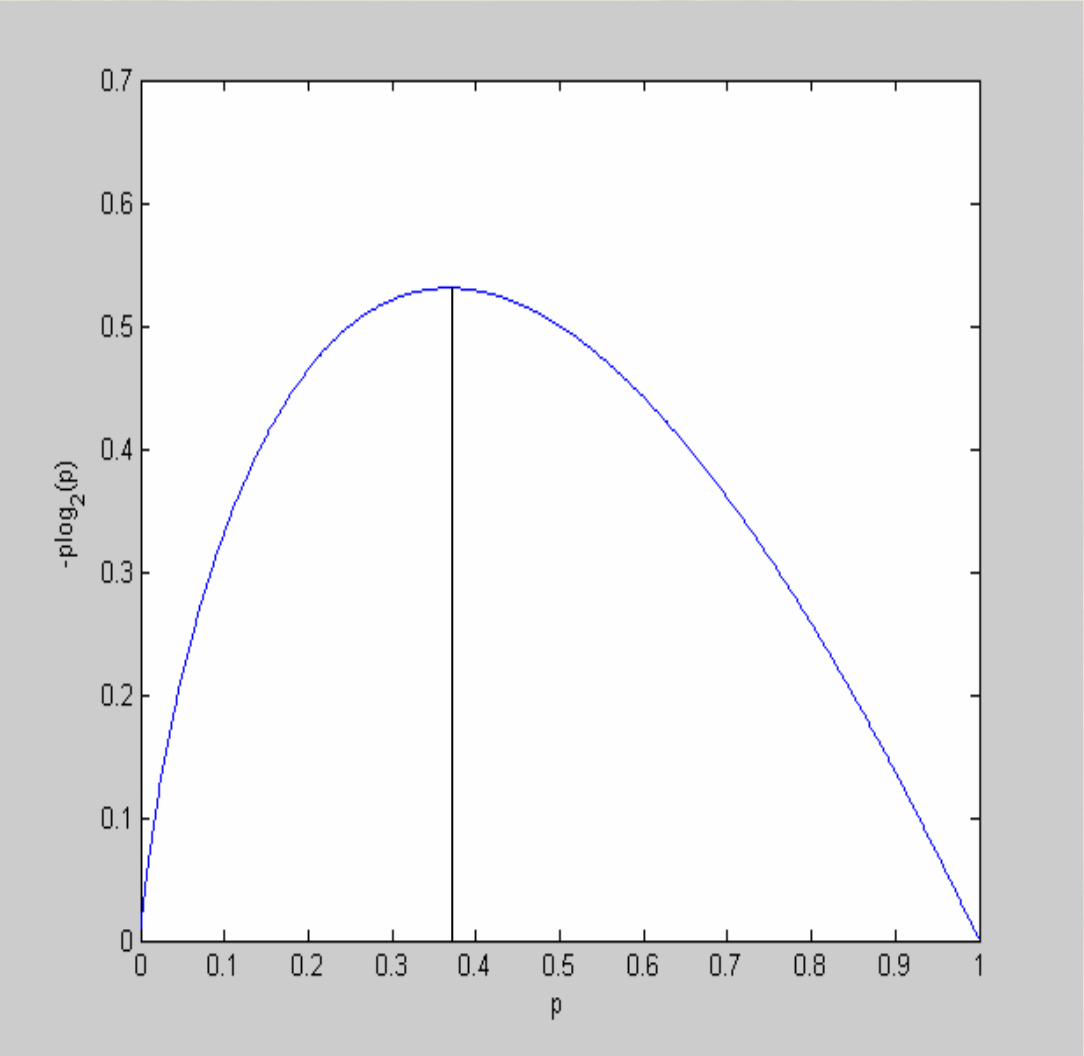
p	$I(p)$	$I_A(p) = p \cdot I(p)$
0	∞	0
1/2	1	1/2
1	0	0

As p evolves from 0 to 1, weighted self-information

$I_A(p) = -p \cdot \log_2 p$ *first increases and then decreases*

<u>Question:</u> Which value of p maximizes $I_A(p)$?

Maximum of Weighted Self-information



$$p = \frac{1}{e}$$

$$\downarrow$$

$$I_A(p) = \frac{1}{e\ln 2}$$

Entropy

- Entropy
  - Average amount of information of a source, more precisely, the average number of bits of information required to represent the symbols the source produces
  - For a source containing *N* independent symbols, its entropy is defined as

$$H = -\sum_{i=1}^{N} P[X_i] \log_2 P[X_i]$$

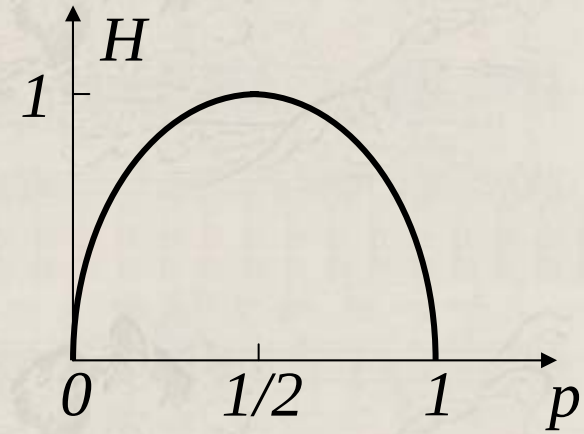
  - Unit of entropy: bits/symbol
  - C. E. Shannon, "A mathematical theory of communication," Bell Systems Technical Journal, 1948

Entropy Example

- Find and plot the entropy of the binary code in which the probability of occurrence for the symbol ***1*** is *p* and for the symbol ***0*** is *1-p*

$$H = -\sum_{i=1}^{2} P[X_i]\log_2 P[X_i]$$

$$= -p\log_2 p - (1-p)\log_2(1-p)$$



$$p = \frac{1}{2} \Rightarrow H = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = \frac{1}{2} + \frac{1}{2} = 1 \text{ bit/symbol}$$

$$p = \frac{1}{4} \Rightarrow H = -\frac{1}{4}\log_2\frac{1}{4} - \frac{3}{4}\log_2\frac{3}{4} = 0.8113 \text{ bits/symbol}$$

$$p = 0 \Rightarrow H = -0\log_2 0 - 1\log_2 1 = 0 \text{ bit/symbol}$$

Entropy Example

- Find the entropy of a DNA sequence containing four equally-likely symbols $\{A,C,T,G\}$

$$H = \left(-\frac{1}{4}\log_2\frac{1}{4}\right)\times 4 = \log_2 4 = 2 \text{ bits/symbol}$$

- *P[A]=1/2; P[C]=1/4; P[T]=P[G]=1/8; H=?*

$$H = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{4}\log_2\frac{1}{4} - \frac{1}{8}\log_2\frac{1}{8} - \frac{1}{8}\log_2\frac{1}{8}$$

$$= \frac{1}{2} + \frac{1}{2} + \frac{3}{8} + \frac{3}{8} = \frac{7}{4} \text{ bits/symbol} < 2 \text{ bits/symbol}$$

- So, how do we design codes to represent DNA sequences?

Fixed-Length Codes

- Properties
  - Use the same number of bits to represent all possible symbols produced by the source
  - Simplify the decoding process
- Examples
  - American Standard Code for Information Interchange (ASCII) code
  - Bar codes
    - One used by the US Postal Service
    - Universal Product Code (UPC) on products in stores
    - Credit card codes

ASCII Code

- ASCII is used to encode and communicate alphanumeric characters for plain text
- 128 common characters: lower-case and upper-case letters, numbers, punctuation marks… 7 bits per character
- First 32 are control characters (for example, for printer control)
- Since a byte is a common structured unit of computers, it is common to use 8 bits per character – there are an additional 128 special symbols
- Example

Character	D	S	P	S	S
Dec. index	68	83	80	83	83
Bin. code	01000100	01010011	01010000	01010011	01010011

ASCII Table

Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH	(start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX	(start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX	(end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT	(end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ	(enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK	(acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL	(bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS	(backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB	(horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT	(vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR	(carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO	(shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI	(shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE	(data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1	(device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2	(device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3	(device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4	(device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK	(negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN	(synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB	(end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN	(cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM	(end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB	(substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC	(escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS	(file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS	(group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS	(record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US	(unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.asciitable.com

www.AsciiTable.com

Variable-Length Codes

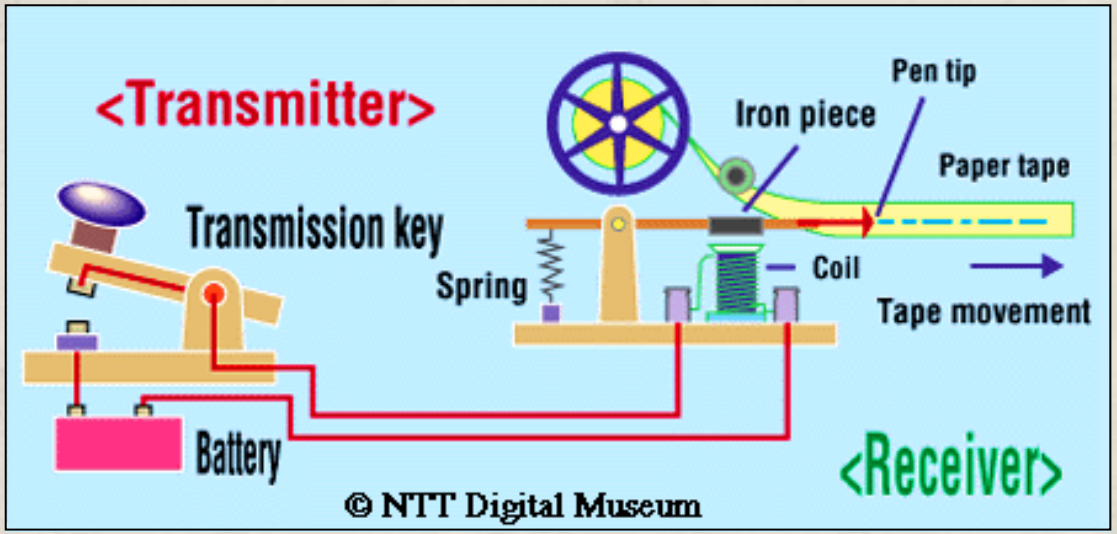
- Main problem with fixed-length codes: inefficiency
- Main properties of variable-length codes (VLC)
  - Use a different number of bits to represent each symbol
  - Allocate shorter-length code-words to symbols that occur more frequently
  - Allocate longer-length code-words to rarely-occurred symbols
  - More efficient representation; good for compression
- Examples of VLC
  - Morse code
  - Shannon-Fano code
  - Huffman code
  - Arithmetic code

Morse Codes & Telegraphy

- Morse codes

E	•	T	-
I	••	M	- -
S	•••	O	- - -
H	••••	CH	- - - -
A	•-	N	-•
U	••-	D	-••
V	•••-	B	-•••
R	•-•	K	-•-
W	•- -	G	- - •
L	•-••	Y	-•- -
P	•- - •	X	-••-
F	••-•	Q	- - •-
J	•- - -	C	-•-•

- "What hath God wrought?", DC – Baltimore, 1844
- Allocate shorter codes for more frequently-occurring letters & numbers
- Telegraph is a binary communication system – dash: **1**; dot: **0**



Issues in VLC Design

- Optimal efficiency
  - How to perform optimal code-word allocation (in an efficiency standpoint) given a particular signal?
- Uniquely decodable
  - No confusion allowed in the decoding process
  - Example: Morse code has a major problem!
    - Message: ***SOS***. Morse code: **000111000**
    - Many possible decoded messages: ***SOS*** or ***VMS***?
- Instantaneously decipherable
  - Able to decipher as we go along without waiting for the entire message to arrive
- Algorithmic issues
  - Systematic design?
  - Simple fast encoding and decoding algorithms?

VLC Example

Symbol	Prob.	FLC	Code 1	Code 2	Code 3	Code 4
A	P[A]=1/2	000	1	1	0	00
B	P[B]=1/4	001	01	10	10	01
C	P[C]=1/8	010	001	100	110	10
D	P[D]=1/16	011	0001	1000	1110	11
E	P[E]=1/16	100	00001	10000	1111	110
Average Length	H=30/16	3	31/16	31/16	30/16	33/16

$$H = -\sum_{i=1}^{N} P[X_i]\log_2 P[X_i]$$

$$= -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{4}\log_2\frac{1}{4} - \frac{1}{8}\log_2\frac{1}{8} - \frac{2}{16}\log_2\frac{1}{16}$$

$$= \frac{1}{2} + \frac{1}{2} + \frac{3}{8} + \frac{8}{16} = \frac{30}{16} = \frac{15}{8} bits/symbol$$

$Code\,1:$

$$E[L] = 1\times\frac{1}{2} + 2\times\frac{1}{4} + 3\times\frac{1}{8} + 4\times\frac{1}{16}$$

$$+5\times\frac{1}{16} = \frac{31}{16} bits/symbol$$

VLC Example

Symbol	Prob.	FLC	Code 1	Code 2	Code 3	Code 4
A	P[A]=1/2	000	1	1	0	00
B	P[B]=1/4	001	01	10	10	01
C	P[C]=1/8	010	001	100	110	10
D	P[D]=1/16	011	0001	1000	1110	11
E	P[E]=1/16	100	00001	10000	1111	110
Average Length	H=30/16	3	31/16	31/16	30/16	33/16

- Uniquely decodable – Self-synchronizing: Code 1, 2, 3. No confusion in decoding
- Instantaneous: Code 1, 3. No need to look ahead.
- Prefix condition = uniquely decodable & instantaneous: no codeword is a prefix of another

Shannon-Fano Code

- Algorithm
  - Line up symbols by decreasing probability of occurrence
  - Divide symbols into 2 groups so that both have similar combined probability
  - Assign **0** to 1st group and **1** to the 2nd
  - Repeat step 2
- Example

H=2.2328 bits/symbol

Symbols	Prob.	Code-word
A	0.35	0 0
B	0.17	0 1
C	0.17	1 0
D	0.16	1 1 0
E	0.15	1 1 1

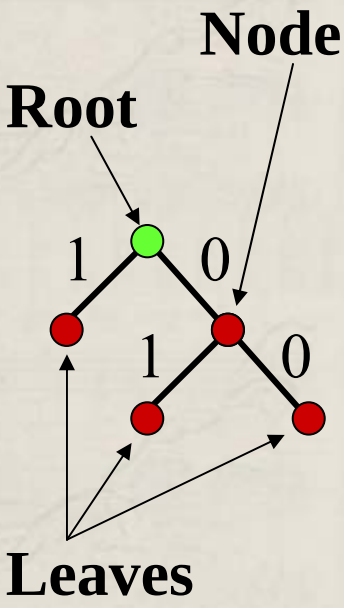
Average code-word length =
0.35 x 2 + 0.17 x 2 + 0.17 x 2
+ 0.16 x 3 + 0.15 x 3
= 2.31 *bits/symbol*

Huffman Code

- Shannon-Fano code [1949]
  - Top-down algorithm: assigning code from most frequent to least frequent
  - VLC, uniquely & instantaneously decodable (no code-word is a prefix of another)
  - Unfortunately not optimal in term of minimum redundancy
- Huffman code [1952]
  - Quite similar to Shannon-Fano in VLC concept
  - Bottom-up algorithm: assigning code from least frequent to most frequent
  - Minimum redundancy when probabilities of occurrence are powers-of-two
  - In JPEG images, DVD movies, MP3 music

Huffman Coding Algorithm

- Encoding algorithm
  - Order the symbols by decreasing probabilities
  - Starting from the bottom, assign **0** to the least probable symbol and **1** to the next least probable
  - Combine the two least probable symbols into one composite symbol
  - Reorder the list with the composite symbol
  - Repeat Step 2 until only two symbols remain in the list
- Huffman tree
  - Nodes: symbols or composite symbols
  - Branches: from each node, 0 defines one branch while 1 defines the other
- Decoding algorithm
  - Start at the root, follow the branches based on the bits received
  - When a leaf is reached, a symbol has just been decoded



Huffman Coding Example

Symbols	Prob.	
A	0.35	
B	0.17	
C	0.17	
D	0.16	1
E	0.15	0

Symbols	Prob.	
A	0.35	
DE	0.31	
B	0.17	1
C	0.17	0

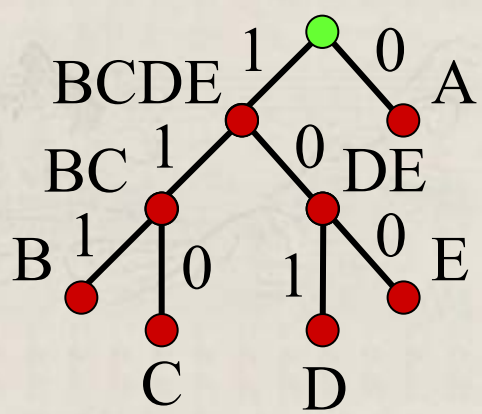
Symbols	Prob.	
A	0.35	
BC	0.34	1
DE	0.31	0

Symbols	Prob.	
BCDE	0.65	1
A	0.35	0

Huffman Codes

A	0
B	111
C	110
D	101
E	100

Huffman Tree



H=2.2328 bits/symbol

Average code-word length = *E[L]* =
0.35 x 1 + 0.65 x 3 = 2.30 bits/symbol

Huffman Coding Example

Symbols	Prob.	
A	1/2	
B	1/4	
C	1/8	
D	1/16	0
E	1/16	1

Symbols	Prob.	
A	1/2	
B	1/4	
C	1/8	0
DE	1/8	1

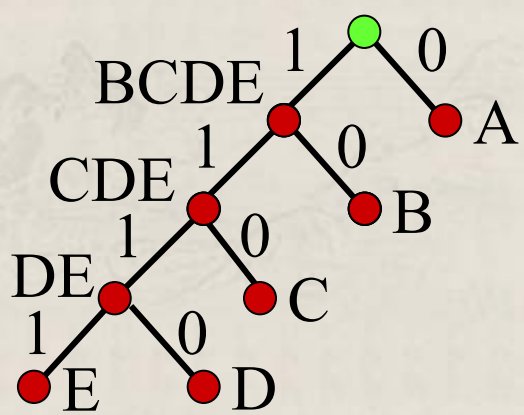
Symbols	Prob.	
A	1/2	
B	1/4	0
CDE	1/4	1

Symbols	Prob.	
A	1/2	0
BCDE	1/2	1

Huffman Codes

A	0
B	10
C	110
D	1110
E	1111

Huffman Tree



Average code-word length = $E[L]$ =

$0.5 \times 1 + 0.25 \times 2 + 0.125 \times 3 + 0.125 \times 4 = 1.875$ bits/symbol $= H$

Huffman Shortcomings

- Difficult to make adaptive to data changes
- Only optimal when $P[X_i] = \frac{1}{2^{k_i}}$
- Best achievable bit-rate = 1 bit/symbol

- Question: What happens if we only have 2 symbols to deal with? A binary source with skewed statistics?
  - Example: *P[0]=0.9375; P[1]=0.0625.*
    H = 0.3373 bits/symbol. Huffman: *E[L] = 1*.
  - One solution: combining symbols!

Extended Huffman Code

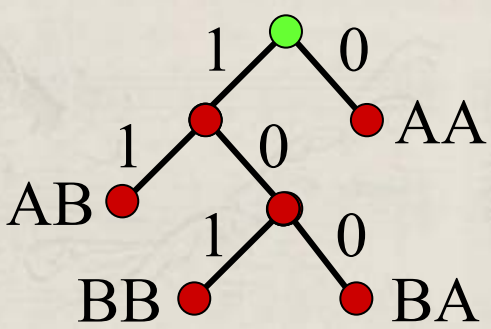
Symbols	Prob.
A=0	15/16
B=1	1/16

H=0.3373 bits/symbol

Symbols	Prob.
AA	225/256
AB	15/256
BA	15/256
BB	1/256

H=0.6746 bits/symbol

Huffman Tree



- Larger grouping yield better performance
- Problems
  - Storage for codes
  - Inefficient & time-consuming
  - Still not well-adaptive

Average code-word length = E[L] = 1 x 225/256 + 2 x 15/256 + 3 x 15/256 + 3 x 1/256 = 1.1836 *bits/symbol* >> *2*

Arithmetic Coding: Main Idea

- Peter Elias in Robert Fano's class!
- Large grouping improves coding performance; however, we do not want to generate codes for all possible sequences
- Wish list
  - a tag (unique identifier) is generated for the sequence to be encoded
  - easy to adapt to statistic collected so far
  - more efficient than Huffman

- <u>Main Idea:</u> tag the sequence to be encoded with a number in the unit interval [0, 1) and send that number to the decoder

- Review: binary representation of fractions
  - $0.75_{10} = 0.5 + 0.25 = 2^{-1} + 2^{-2} = 0.11_2$
  - $0.384765625_{10} = 2^{-2} + 2^{-3} + 2^{-7} + 2^{-9} = 0.011000101_2$

Coding Example

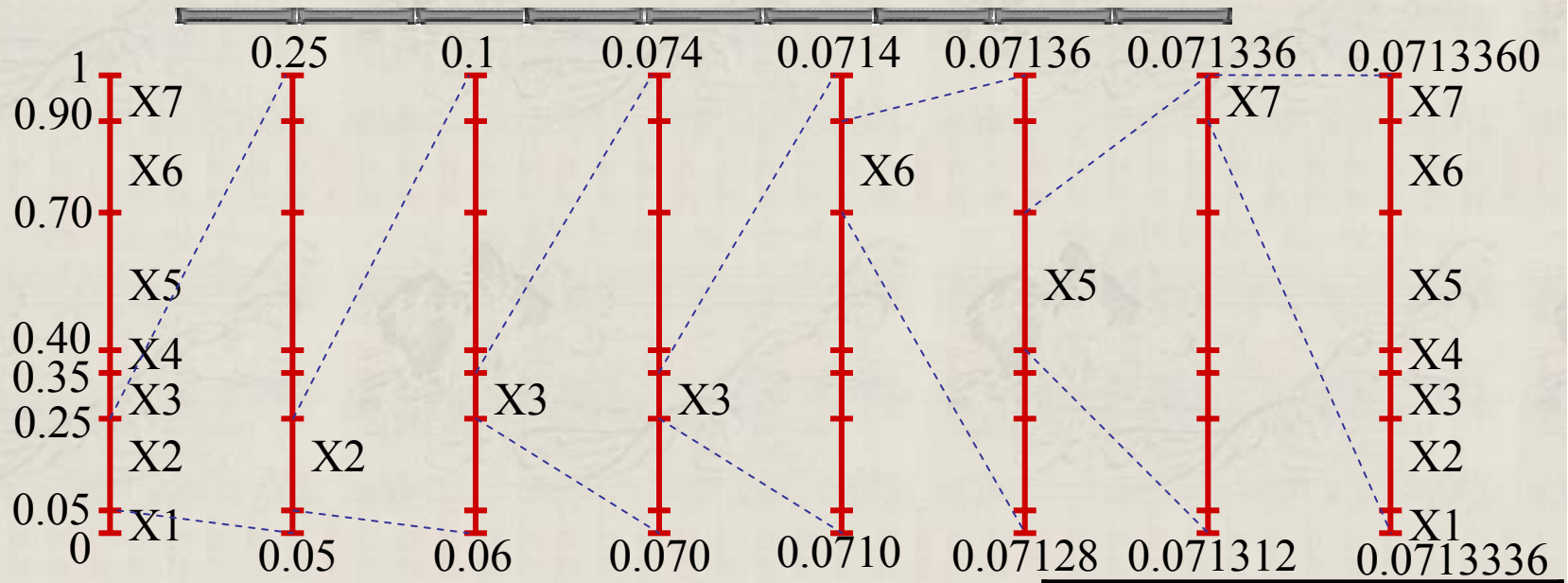
Symbol	Probability	Huffman Code
X1	0.05	10101
X2	0.2	01
X3	0.1	100
X4	0.05	10100
X5	0.3	11
X6	0.2	00
X7	0.1	1011

String to encode: X2 X2 X3 X3 X6 X5 X7

Huffman: 01 01 100 100 00 11 1011

18 bits

Arithmetic Encoding Process



String to encode: X2 X2 X3 X3 X6 X5 X7

range = high – low

new_high = low + range x subinterval_high

new_low = low+range x subinterval_low

Final interval = [0.0713336,0.0713360)

16 bits

Send to decoder: 0.07133483886719

$$2^{-4}+2^{-7}+2^{-10}+2^{-15}+2^{-16}=0.0001001001000011_2$$

Sym	Prob	Huffman
X1	0.05	10101
X2	0.2	01
X3	0.1	100
X4	0.05	10100
X5	0.3	11
X6	0.2	00
X7	0.1	1011

Arithmetic Decoding Process

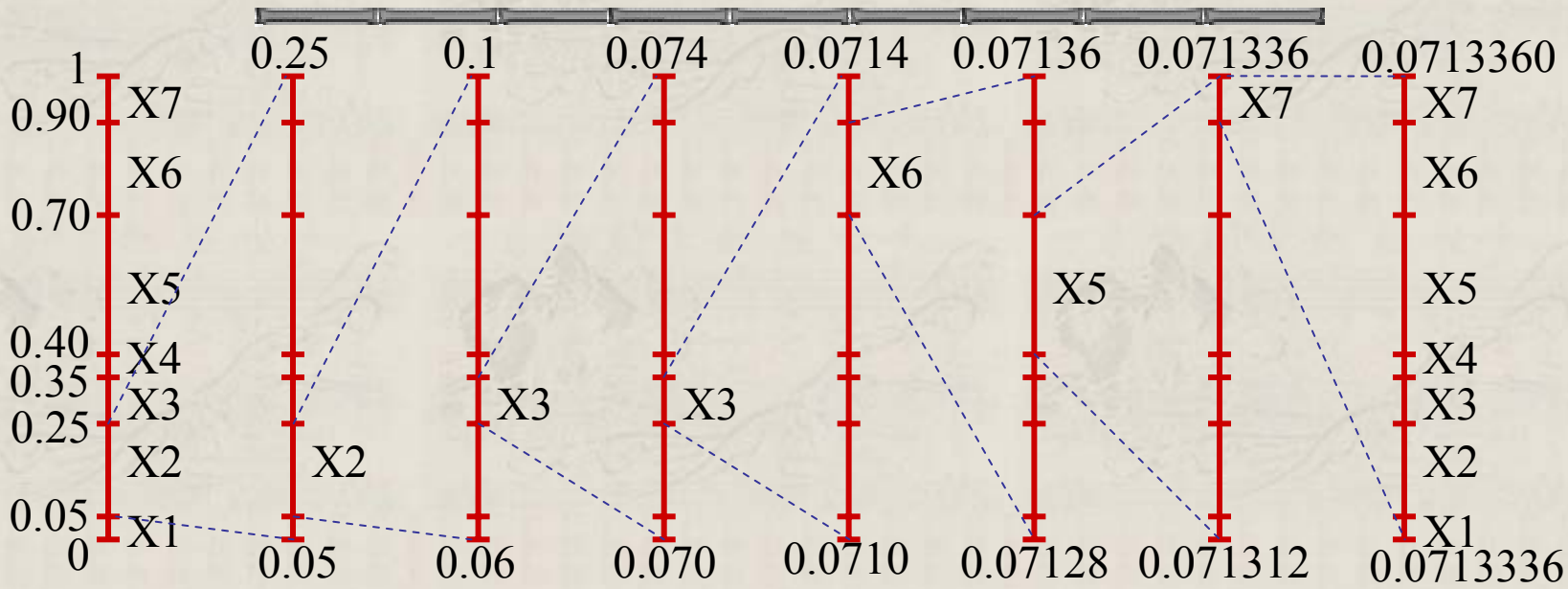
- low=0; high=1; range=high – low
- REPEAT
  - Find index i such that

$$\text{subinterval_low} \le \frac{value - low}{range} \le \text{subinterval_high}$$

  - OUTPUT SYMBOL
  - high = low + range x subinterval_high
  - low = low + range x subinterval_low
  - range = high – low

  UPDATE
- UNTIL END

Arithmetic Decoding Example



$$low = 0; high = 1; range = 0$$

$$0.05 \le \frac{0.071334... - 0}{1} \le 0.25 \Rightarrow X_2$$

$$high = 0 + 1 \times 0.25 = 0.25$$

$$low = 0 + 1 \times 0.05 = 0.05$$

$$0.05 \le \frac{0.071334... - 0.05}{0.20} = 0.1067 \le 0.25 \Rightarrow X_2$$

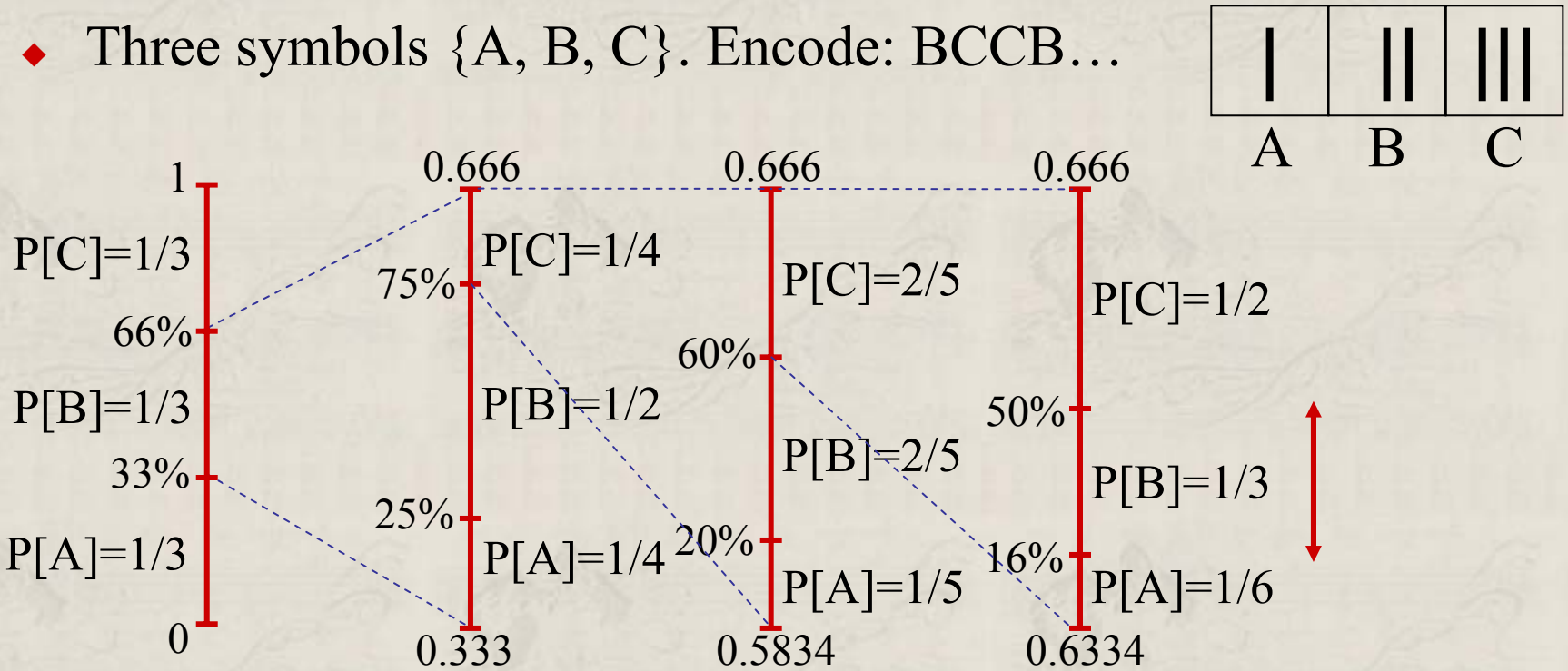
$$high = 0.05 + 0.2 \times 0.25 = 0.1$$

$$low = 0.05 + 0.2 \times 0.05 = 0.06$$

$$0.25 \le \frac{0.071334... - 0.06}{0.04} = 0.2834 \le 0.35 \Rightarrow X_3...$$

Adaptive Arithmetic Coding

- Three symbols {A, B, C}. Encode: BCCB…



Final interval = [0.6390, 0.6501)

Decode?

$$output = 0.6396_{10} =$$

$$2^{-1} + 2^{-3} + 2^{-7} + 2^{-8} + 2^{-9} + 2^{-10} = 0.1010001111_2$$

Arithmetic Coding: Notes

- Size of final interval $= \prod_i P[X_i]$
- Symbol X_i of probability $P[X_i]$ contributes $\log_2 P[X_i]$ bits to the output
- Arithmetic coding approaches entropy!
- Near-optimal: finite-precision arithmetic, a whole number of bits or bytes must be sent
- Implementation issues:
  - Incremental output: should not wait until the end of the compressed bit-stream; prefer incremental transmission scheme
  - Prefer integer implementations by appropriate scaling

Run-Length Coding

- Main idea
  - Encoding long runs of a single symbol by the length of the run
- Properties
  - A lossless coding scheme
  - Our first attempt at inter-symbol coding
  - Really effective with transform-based coding since the transform usually produces long runs of *insignificant* coefficients
  - Run-length coding can be combined with other entropy coding techniques (for example, run-length and Huffman coding in JPEG)

Run-Length Coding

- Example: How do we encode the following string?

$$14 \quad 0 \quad 0 \quad 5 \quad 0 \quad -3 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad \overbrace{0 \quad \cdots \quad 0}^{14\text{ zeros}} \quad -1 \quad \overbrace{0 \quad \cdots \quad 0}^{37\text{ zeros}}$$

- Run-length coding:

(run-length, size) binary amplitude value

number of consecutive zeros before current non-zero symbol

number of bits needed to encode this non-zero symbol

actual value of the non-zero symbol in binary

(0,4) 14 (2,3) 5 (1,2) -3 (5,1) 1 (14,1) -1 (0,0)

Run-Length Coding

$$14 \quad 0 \quad 0 \quad 5 \quad 0 \quad -3 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad \overbrace{0 \quad \cdots \quad 0}^{\text{14 zeros}} \quad -1 \quad \overbrace{0 \quad \cdots \quad 0}^{\text{37 zeros}}$$

(run-length, size) binary value

always **1**, no need to encode

sign bit	~~MSB~~	…	LSB

0: positive
1: negative

$$\text{binary of } S = \begin{cases} S - 2^{size-1}, & S > 0 \\ |S|, & \text{otherwise} \end{cases}$$

(0,4) 14 (2,3) 5 (1,2) -3 (5,1) 1 (14,1) -1 (0,0)

Huffman or arithmetic coding

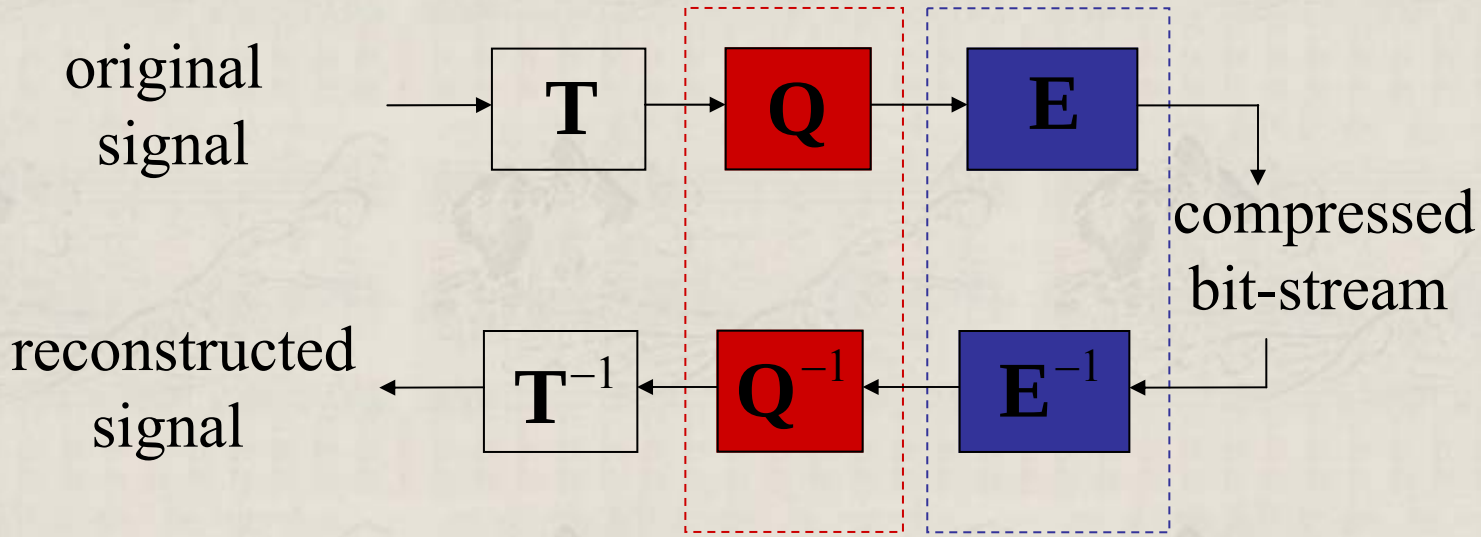
raw binary

4	000
3	01
2	00
1	0
-1	1
-2	10
-3	11
-4	100

Quantization

Trac D. Tran
ECE Department
The Johns Hopkins University
Baltimore, MD 21218

Reminder



Quantization

Information theory
VLC
Huffman
Arithmetic
Run-length

Quantization

- Entropy coding techniques
  - Perform lossless coding
  - No flexibility or trade-off in bit-rate versus distortion
- Quantization
  - Lossy non-linear mapping operation: a range of amplitude is mapped to a unique level or codeword
  - Approximation of a signal source using a finite collection of discrete amplitudes
  - Controls the rate-distortion trade-off
  - Applications
    - A/D conversion
    - Compression

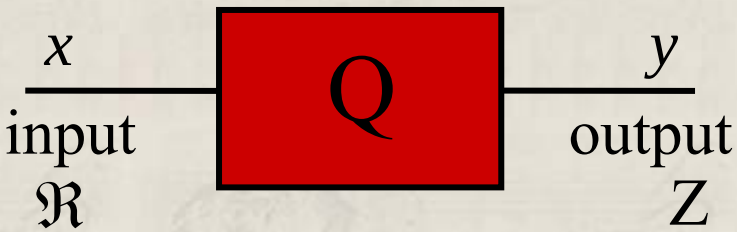
Typical Quantizer

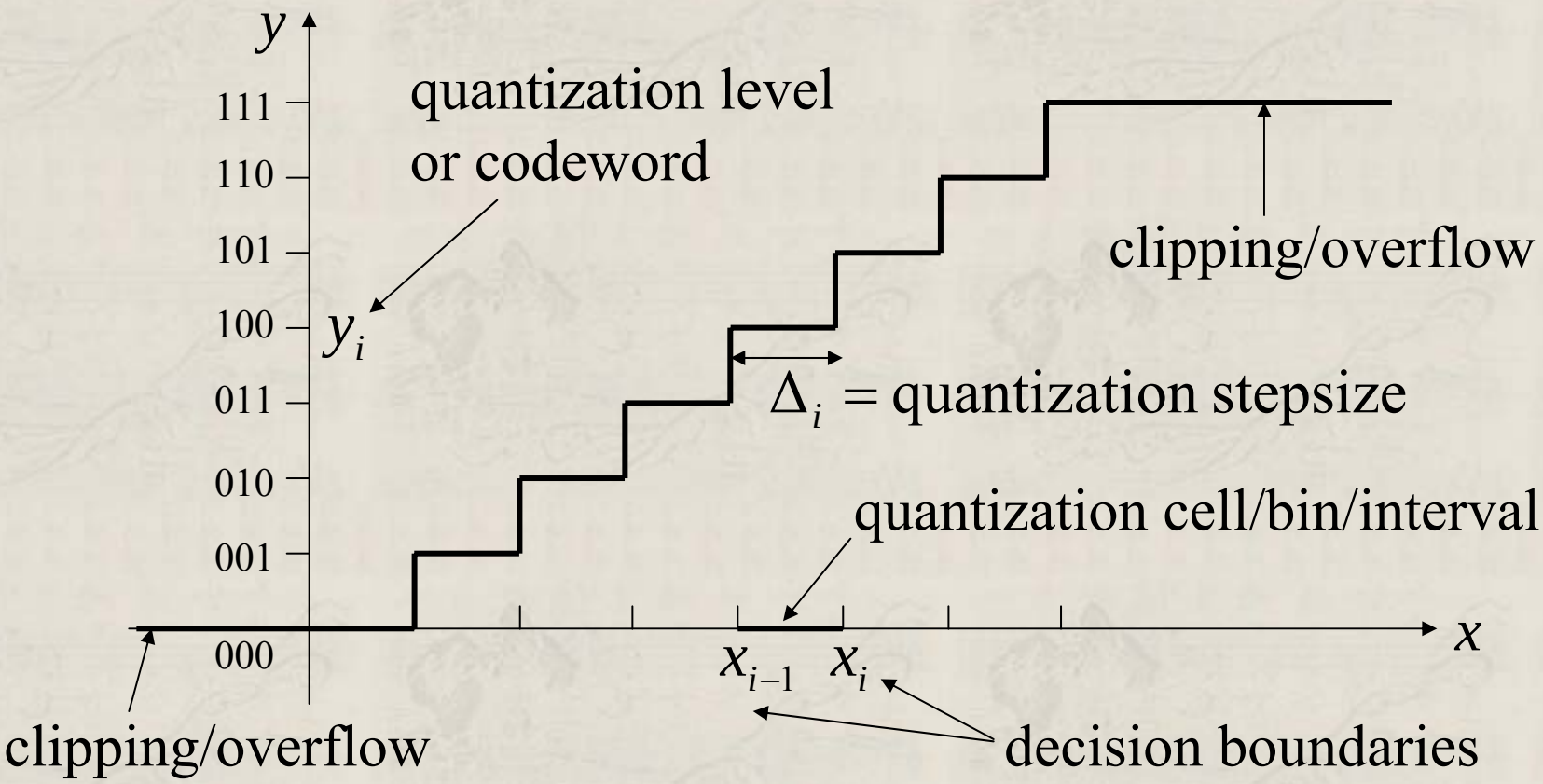
$$x \in I_i = [x_{i-1}, x_i)$$

$$i.e.,\ I_i = \{x \mid x_{i-1} \le x < x_i\}$$

$$\Rightarrow y = y_i$$

Forward Quantizer





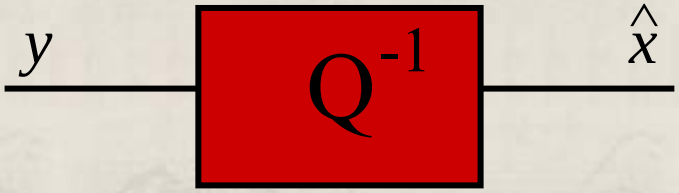
Typical Inverse Quantizer

- Typical reconstruction

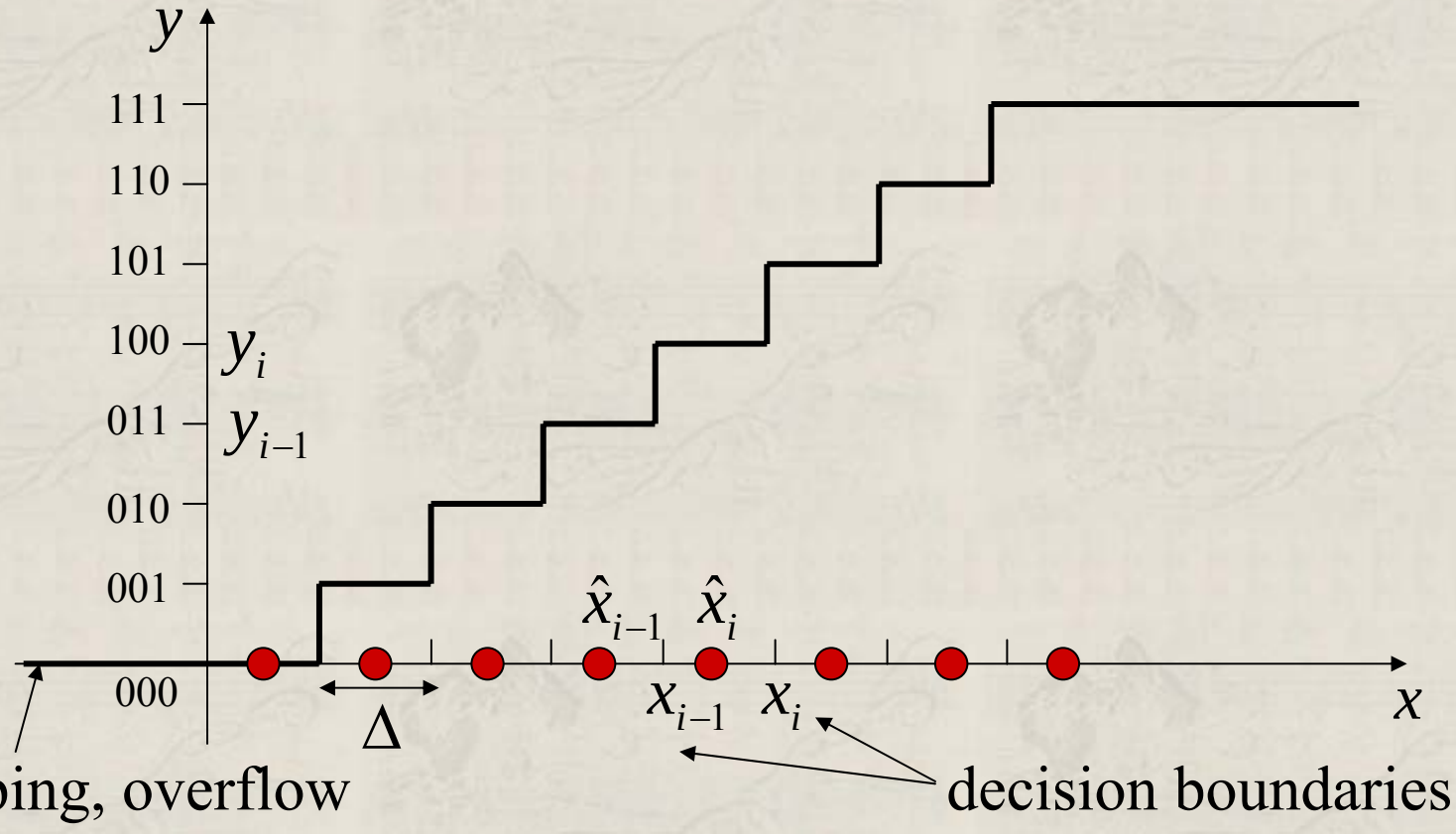
$$\hat{x}_i = \frac{x_i + x_{i-1}}{2}$$

- Quantization error

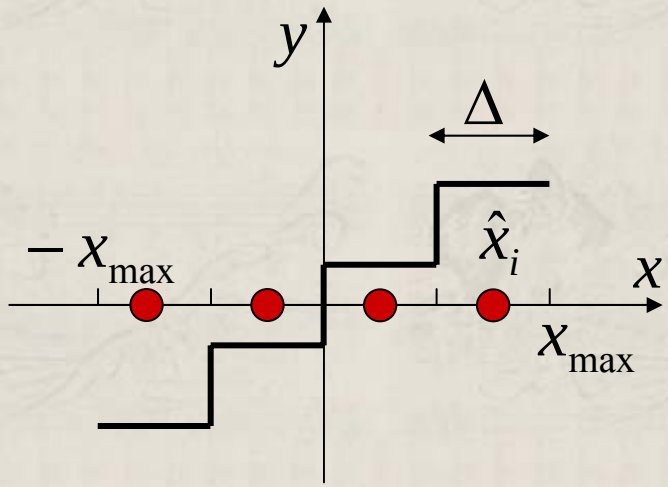
$$q = \hat{x} - x$$

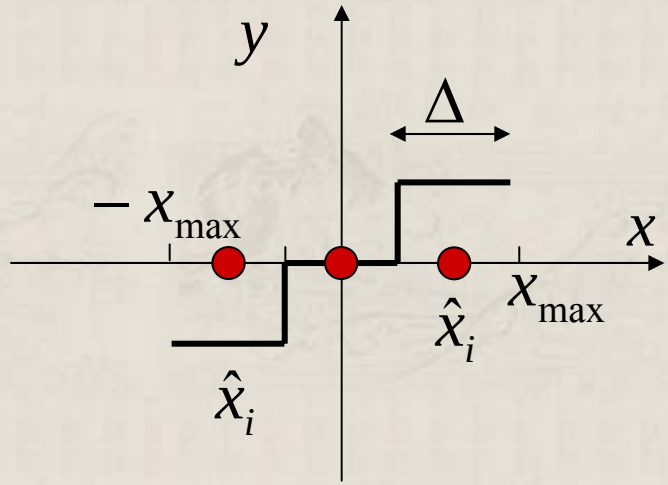


Inverse Quantizer



Mid-rise versus Mid-tread





Uniform Midrise Quantizer

- Popular in ADC
- For a *b*-bit midrise

$$\Delta = \frac{2x_{max}}{2^b}$$

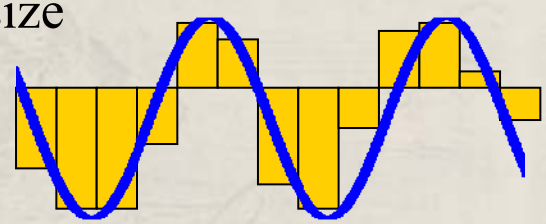
Uniform Midtread Quantizer

- Popular in compression
- For a *b*-bit midtread

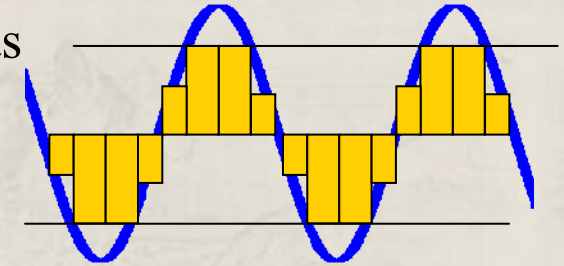
$$\Delta = \frac{2x_{max}}{2^b - 1}$$

Quantization Errors

- Approximation error
  - Lack of quantization resolution, too few quantization levels, too large quantization step-size
  - Causes staircase effect
  - Solution: increases the number of quantization levels, and hence, increase the bit-rate



- Clipping error
  - Inadequate quantizer range limits, also known as overflow
  - Solution
    - Requires knowledge of the input signal
    - Typical practical range for a zero-mean signal

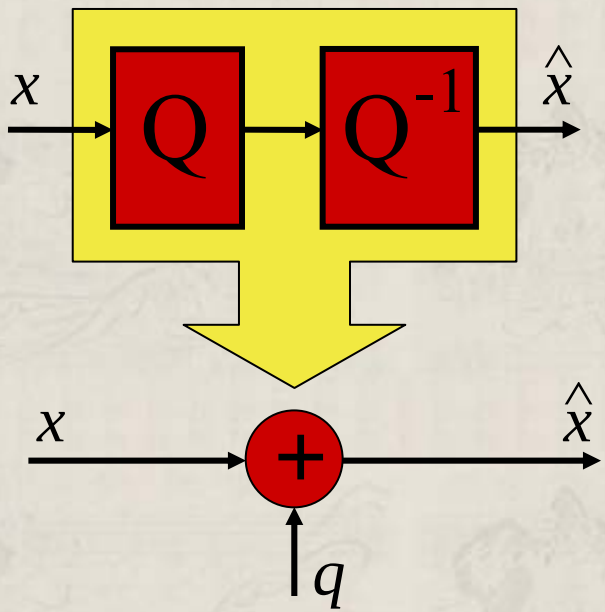


$$x_{\max} = 4x_{RMS}$$

$$x_{\min} = -4x_{RMS}$$

$$x_{RMS} = \sqrt{\frac{1}{T}\int_0^T x^2(t)dt}$$

Quantization: Error Model



- Assumptions:

  x RV with PDF $f_X(x)$

  y RV; $\hat{x}$ RV

  q 0-mean RV independent of x

- Quantization error:

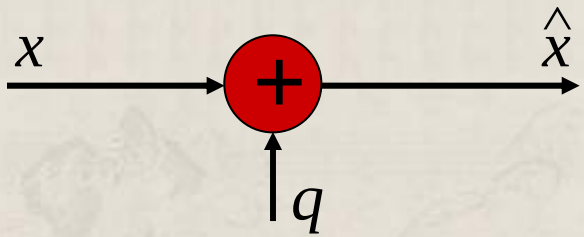
$$\hat{x} = x + q \Rightarrow q = \hat{x} - x$$

- Mean-squared distortion measure:

$$D(x,\hat{x}) = E[(\hat{x}-x)^2] = E[\underbrace{(x-\hat{x})^2}_{\text{function of } x}]$$

$$= E[q^2] = \text{quantization error variance}$$

Quantization Error Variance



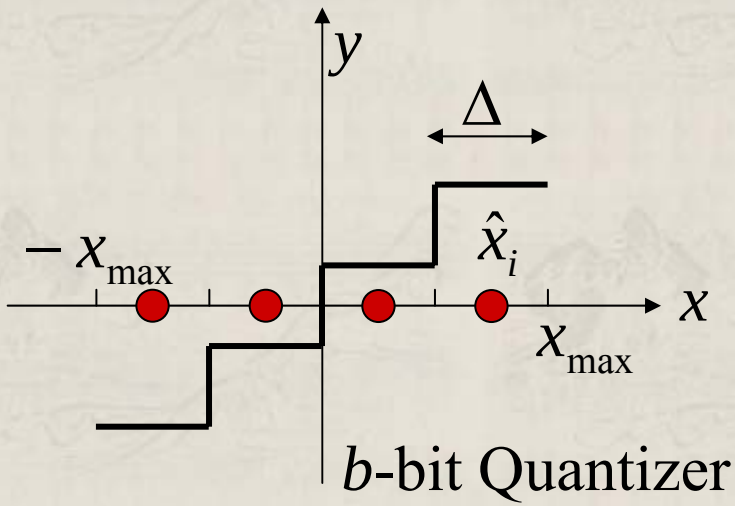
$$D(\hat{x}, x) = E[(x - \hat{x})^2] = \int_{-\infty}^{\infty} (x - \hat{x})^2 f_X(x) dx$$

$$= \int_{x_1}^{x_2} (x - \hat{x}_1)^2 f_X(x) dx + \int_{x_2}^{x_3} (x - \hat{x}_2)^2 f_X(x) dx + \cdots$$

$$= \boxed{\sum_{k=1}^{L} \int_{x_k}^{x_{k+1}} (x - \hat{x}_k)^2 f_X(x) dx}$$

would like to minimize

Uniform Quantization – Bounded Input

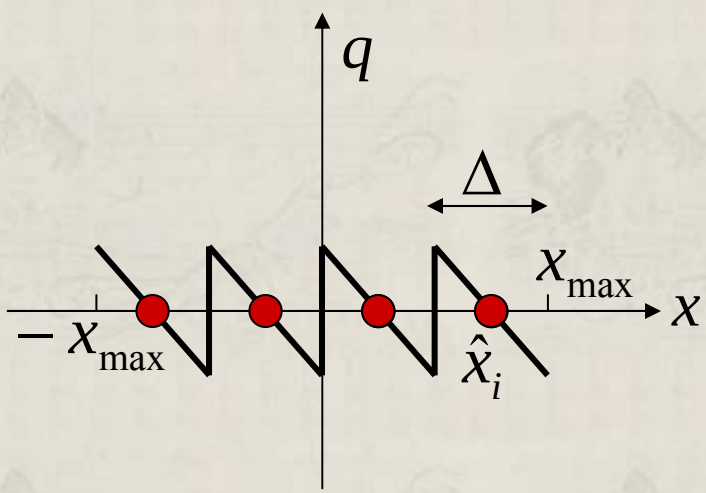


b-bit Quantizer

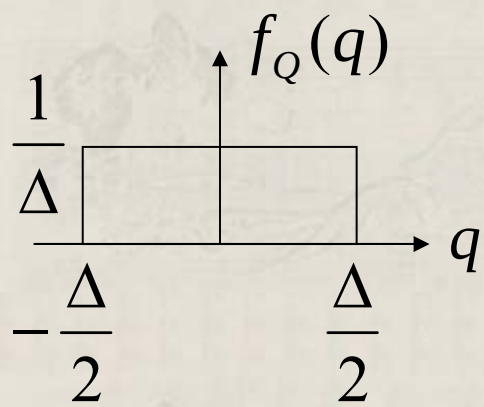
Bounded input : $-x_{\max} \le x \le x_{\max}$

Center reconstruction : $\hat{x}_i = \dfrac{x_i + x_{i-1}}{2}$

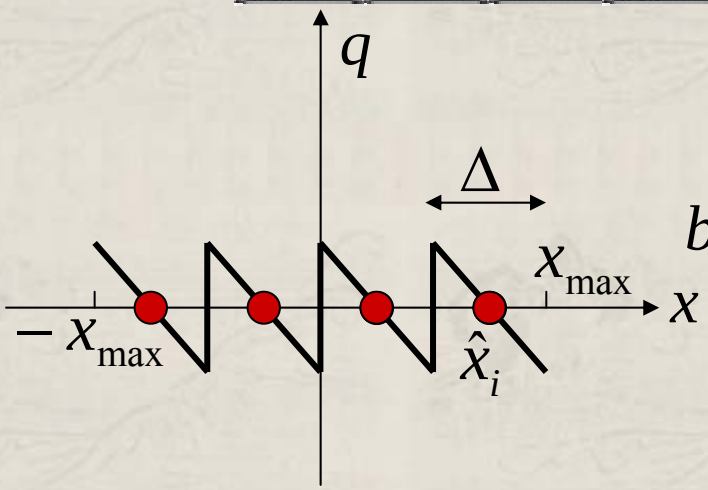
Error bound : $-\dfrac{\Delta}{2} \le q \le \dfrac{\Delta}{2}$



high bit-rate assumption



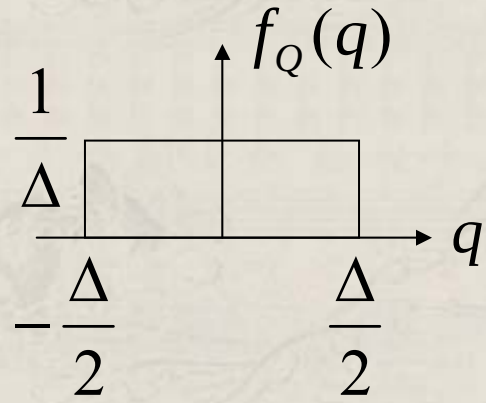
Uniform Quantization – Bounded Input



$$\Delta = \frac{2x_{\max}}{2^b}$$

b-bit quantizer

high bit-rate assumption



$$D(\hat{x}, x) = E[q^2] = \int_{-\infty}^{\infty} q^2 f_Q(q)dq = ?$$

Increase b by 1 bit/symbol $\Rightarrow$ double L

$\Rightarrow$ reduce Δ by a factor of 2

Signal-to-Noise Ratio

- Definition of SNR in decibel (dB)

$$SNR_{dB} = 10\log_{10}\frac{\sigma_x^2}{\sigma_q^2}$$

power of the signal

power of the noise

- For quantization noise

$$SNR_{dB} = 10\log_{10}\frac{12\sigma_x^2}{\Delta^2} = 10\log_{10}12\sigma_x^2 - 20\log_{10}\Delta$$

Suppose that we now add 1 more bit to our Q resolution:

$$\Delta' = \frac{\Delta}{2} \quad \Rightarrow SNR_{dB}^{'} = 10\log_{10}12\sigma_x^2 - 20\log_{10}\Delta'$$

$$\Rightarrow SNR_{dB}^{'} = 10\log_{10}12\sigma_x^2 - 20\log_{10}\Delta + 20\log_{10}2$$

$$= SNR_{dB} + 20\log_{10}2$$

$$\Rightarrow SNR_{dB}^{'} \approx SNR_{dB} + 6dB$$

Example

Design a 3-bit uniform quantizer for a signal with range [*0,128*]

- Maximum possible number of levels: $L = 2^3 = 8$
- Quantization stepsize: $\Delta = \dfrac{x_{\max} - x_{\min}}{L} = \dfrac{128}{8} = 16$
- Quantization levels: $y_i = \{0,1,2,3,4,5,6,7\}$
- Reconstruction levels: $\hat{x}_i = \{8,24,40,56,72,88,104,120\}$
- Maximum quantization error: $|q| \le 8$

Example of Popular Quantization

- Round

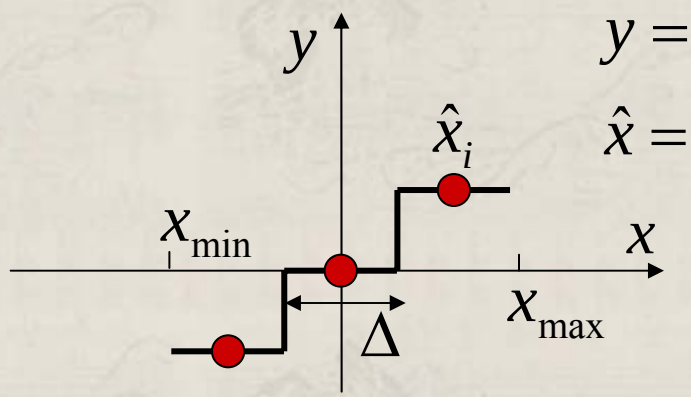
  $y = round(x) =$ nearest integer to x

- Floor

  $y = floor(x) = \lfloor x \rfloor =$ largest integer smaller than x

- Ceiling

  $y = ceil(x) = \lceil x \rceil =$ smallest integer larger than x



$$y = round\left(\frac{x}{\Delta}\right) = \left\lfloor \frac{x}{\Delta} + 0.5 \right\rfloor$$

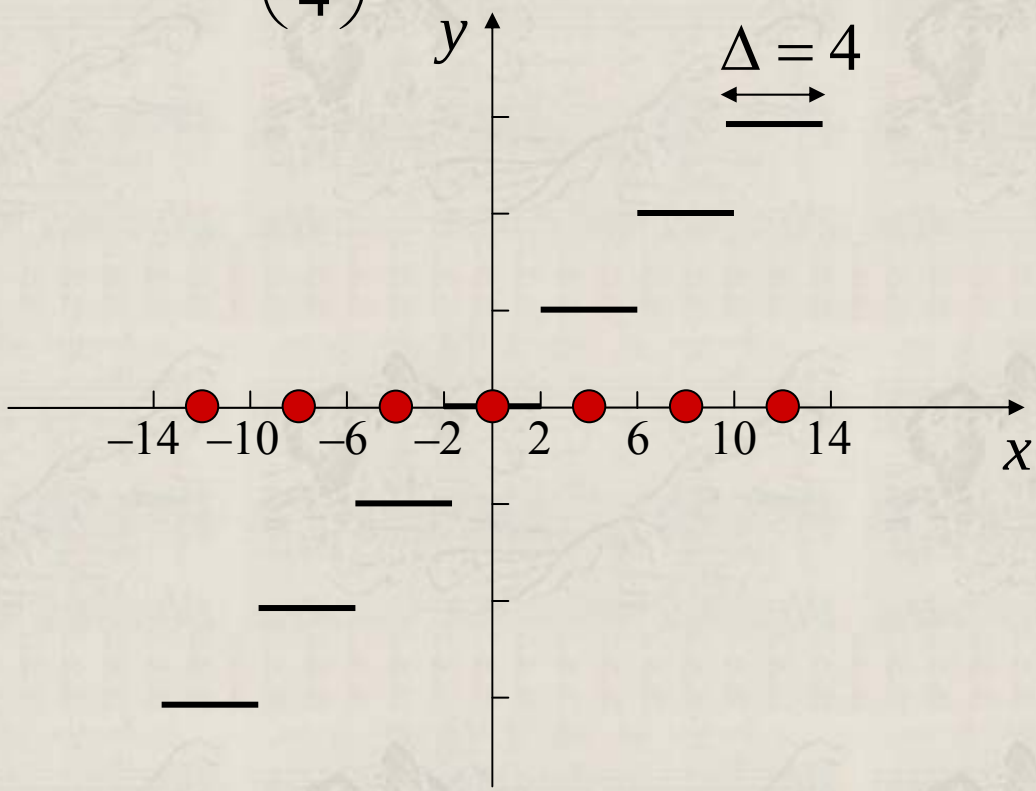
$$\hat{x} = \Delta \times y$$

Uniform midtread quantizer from Round and Floor

Quantization from Rounding

Bounded input : $x \in (-16, 16)$

$\Delta = 4;\ \mathrm{y} = \mathrm{round}\left(\frac{x}{4}\right);\ \hat{x} = 4y$



Uniform Quantizer, *step-size=4*

Optimal Scalar Quantization

- Problem Statement:

  x RV with known PDF $f_X(x)$

  Find $x_k \& \hat{x}_k$ such that $D(\hat{x},x)$ is minimized

- Optimal Encoder for a Given Decoder:

  Given $\hat{x}_k$, find x_k such that $D(\hat{x},x)$ is minimized

$$\text{Minimize} \quad D(\hat{x},x) = \sum_{k=1}^{L} \int_{x_k}^{x_{k+1}} (x - \hat{x}_k)^2 f_X(x)dx \quad \text{w.r.t} \quad x_k$$

- Notes:
  - Non-uniform quantizer under consideration
  - Reconstruction can be anywhere, not necessarily the center of the interval

Optimal Scalar Quantization

$$\text{Minimize} \quad D(\hat{x}, x) = \sum_{k=1}^{L} \int_{x_k}^{x_{k+1}} (x - \hat{x}_k)^2 f_X(x) dx \quad \text{w.r.t} \;\; x_k$$

Optimal Scalar Quantization

- Optimal Decoder for a Given Encoder:

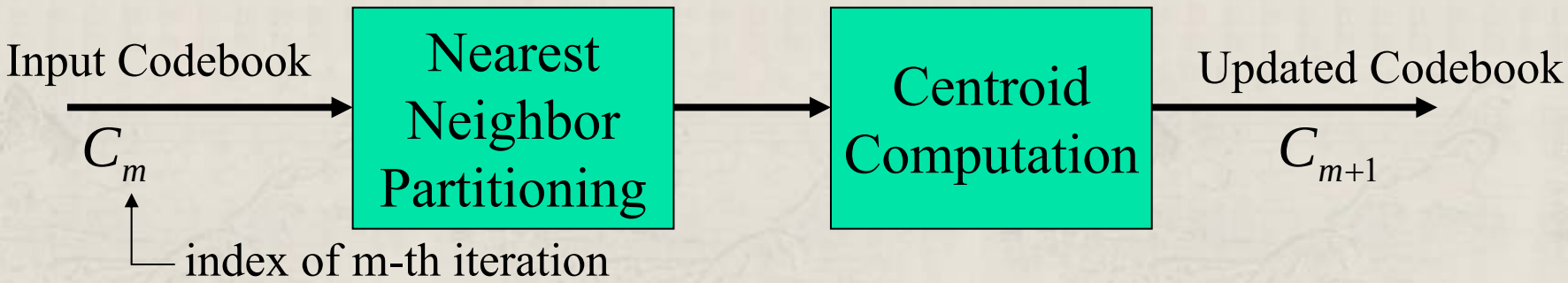
  Given x_k, find $\hat{x}_k$ such that $D(\hat{x},x)$ is minimized

$$\text{Minimize} \quad D(\hat{x},x) = \sum_{k=1}^{L} \int_{x_k}^{x_{k+1}} (x - \hat{x}_k)^2 f_X(x)dx \quad \text{w.r.t} \ \hat{x}_k$$

Lloyd-Max Quantizer

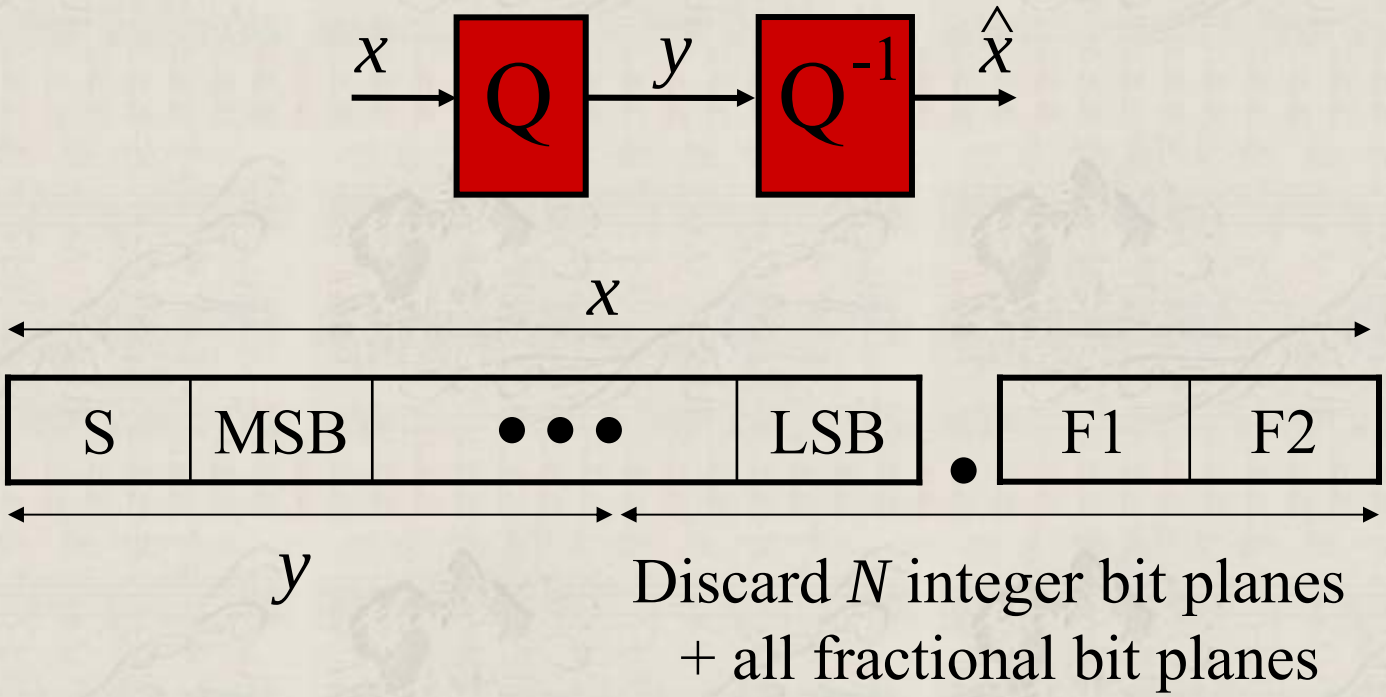
- Unfortunately, we need $\hat{x}_k$ to solve for x_k and x_k to solve for $\hat{x}_k$!

- Main idea [Lloyd 1957] [Max 1960]
  - solving these 2 equation iteratively until D converges



- Assumptions
  - Input PDF is known and stationary
  - Entropy has not been taken into account

Embedded Quantization

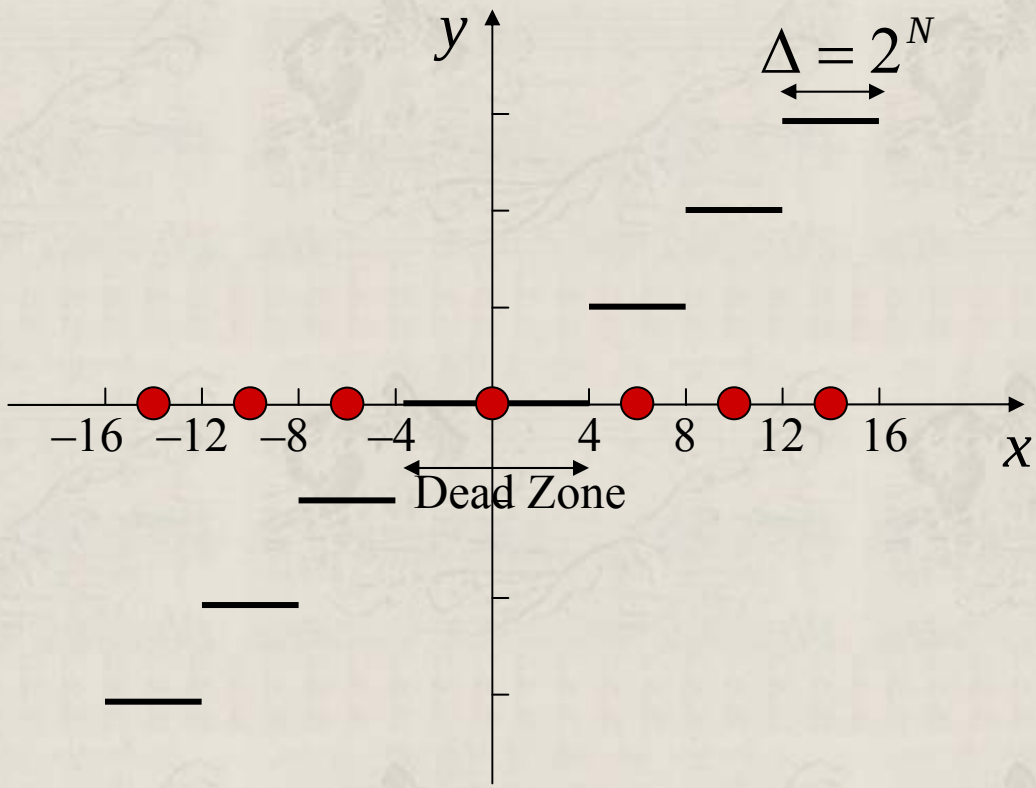


- Also called bit-plane quantization, progressive quantization
- Most significant information is transmitted first
- JPEG2000 quantization strategy

Embedded Forward Quantization

Bounded input : $x \in (-16, 16)$

N = 2 least significant bit planes discarded



Embedded Quantizer, *N=2*

Embedded Inverse Quantization

1	1	1	1
0	X	0	0
1	X	X	1
1	X	X	X
0	X	X	X
Original symbol $x = 22$	Truncate 4 bit planes	Receive 1 refinement bit	Receive 2 refinement bits
	Range=[16, 32)	Range=[16, 24)	Range=[20, 24)
	$\hat{x} = 24$	$\hat{x} = 20$	$\hat{x} = 22$
		$= 24 - 4$	$= 20 + 2$

- N-bit-plane truncation = scalar quantization with $\Delta = 2^N$

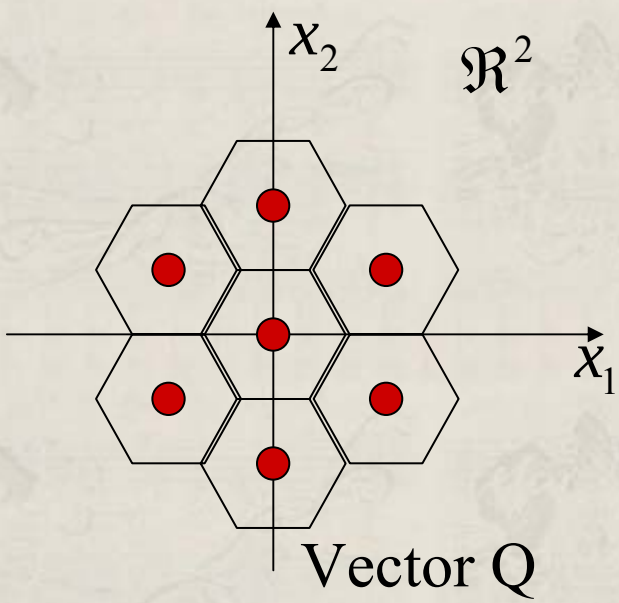
Vector Quantization

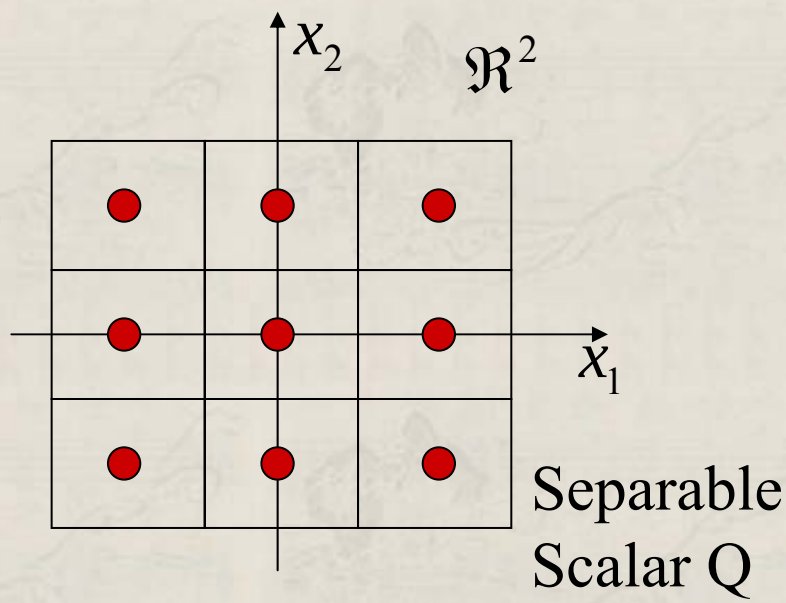
- n-dimensional generalization of scalar quantizer
- $Q: \ \Re^n \rightarrow C$

n-dimensional input vectors

codebook, containing code-vectors or codewords

- Nearest neighbor and centroid rule still apply





Summary

- Introduction to image/video signals
  - Visual information is everywhere in our everyday life
  - Efficient representation (compression) of image/video facilitates visual information storage, archival, communications and even processing
  - Compression is achievable since visual data contains a lot of redundancy, both spatially and temporally
- Variable-length coding
  - Has its roots from information theory
  - Provides effective lossless coding capability
  - Huffman coding & arithmetic coding achieve bit rates close to entropy – the lowest achievable bit rate
- Quantization
  - Allows lossy signal coding and provides trade-off between bit rate and reconstruction quality