INFORMATION THEORY & GENERIC COMPRESSION TECHNIQUES

TOPICS TO BE COVERED

Need for Compression

Data Sources and Information Representation

Lossless Compression Techniques

Lossy Compression Techniques

NEED FOR COMPRESSION

Multimedia information has to be stored and transported efficiently.

Multimedia information is bulky –
Interlaced HDTV: 1080x1920x30x12 = 745 Mb/s!

Use technologies with more bandwidth
- Expensive
- Research

Find ways to reduce the number of bits to transmit without compromising on the "information content"
- Compression Technology
- Reliable Transmission

INFORMATION THEORY

Pioneered primarily by Claude Shannon

Information Theory deals with two sub topics

Source Coding **– How can a communication system efficiently transmit the information that a source produces? Relates to Compression**

Channel Coding **– How can a communication system achieve reliable communication over a noisy channel? Relates to minimizing Transmission Error**

TYPES OF COMPRESSION

Lossless (entropy coding)

- **Does not lose information - the signal is *perfectly* reconstructed after decompression**
- **Produces a variable bit-rate**
- **It is not guaranteed to actually reduce the data size**

Lossy

- **Loses some information - the signal *is not perfectly* reconstructed after decompression**
- **Produces any desired *constant* bit-rate**

MODEL INFORMATION AT THE SOURCE

Model data at the Source as a Stream of Symbols - This defines the "*Vocabulary*" of the source.

Each symbol in the vocabulary is represented by bits

If your vocabulary has N symbols, each symbol represented with $log_2 N$ bits.

- **Speech - 16 bits/sample: $N = 2^{16} = 65{,}536$ symbols**
- **Color Image: 3x8 bits/sample: $N = 2^{24} = 17 \times 10^6$ symbols**
- **8x8 Image Blocks: 8x64 bits/block: $N = 2^{512} = 10^{77}$ symbols**

LOSSLESS COMPRESSION

Lossless compression techniques ensure no loss of data after compression/decompression.

Coding: "Translate" each symbol in the vocabulary into a "*binary codeword*". Codewords may have *different* binary lengths.

Example: You have 4 symbols *(a, b, c,* d). Each in binary may be represented using 2 bits each, but coded using a different number of bits.

***a* (00) $\Rightarrow$ 000**
***b* (01) $\Rightarrow$ 001**
***c* (10) $\Rightarrow$ 01**
***d* (11) $\Rightarrow$ 1**

Goal of Coding is to *minimize* the average symbol length

AVERAGE SYMBOL LENGTH

Symbol Length *l(i)* = binary length of *i* th symbol

Source emits *M* symbols (one every *T* seconds)

Symbol *i* has been emitted *m(i)* times: *M* = $\sum_{i=1}^{i=N} m(i)$

Number of bits been emitted: *L* = $\sum_{i=1}^{i=N} m(i)l(i)$

Average length per symbol: $\bar{L}$ = $\sum_{i=1}^{i=N} m(i)l(i) \Big/ M$

Average bit rate: $\bar{L}$ / *T*

MINIMUM AVERAGE SYMBOL LENGTH

Main goal is to minimize the number of bits being transmitted ⇔ mimimize the average symbol length.

Basic idea for reducing the average symbol length: assign *Shorter Codewords* to symbols that appear more frequently, *Longer Codewords* to symbols that appear less frequently

Theorem: "the *Average Binary Length* of the encoded symbols is always *greater than* or *equal to* the *entropy H* of the source"

What is the *entropy* of a source of symbols and how is it computed?

SYMBOL PROBABILITIES

Probability P(i) **of a symbol: number of times it can occur in the transmission (also relative frequency) and is defined as : P(i) = m(i)/M**

From Probability Theory we know

$$0 \leq P(i) \leq 1 \quad \& \quad \sum_{i=1}^{i=N} P(i) = 1$$

Average symbol length is defined as

$$\sum_{i=1}^{i=N} m(i)l(i) \Big/ M = \sum_{i=1}^{i=N} \left(m(i) \big/ M \right) l(i) = \sum_{i=1}^{i=N} P(i)l(i)$$

ENTROPY

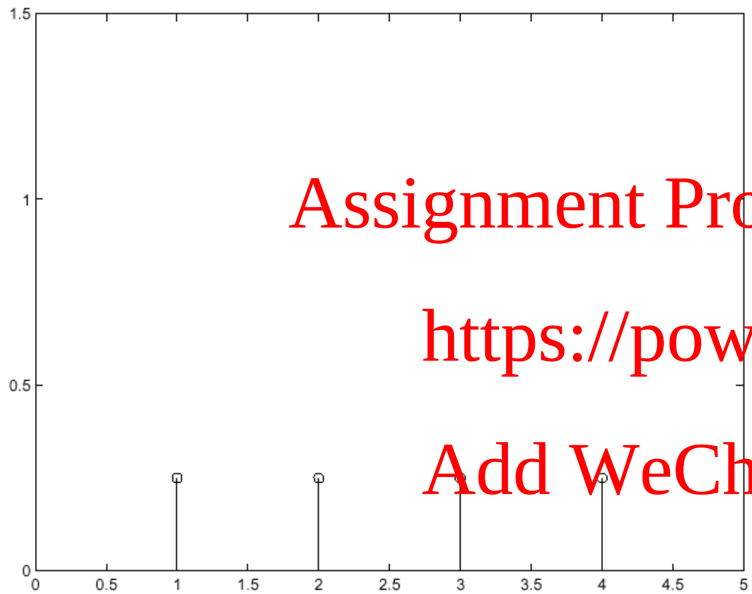
Entropy is defined as $$H = -\sum_{i=1}^{i=N} P(i)\log_2 P(i)$$

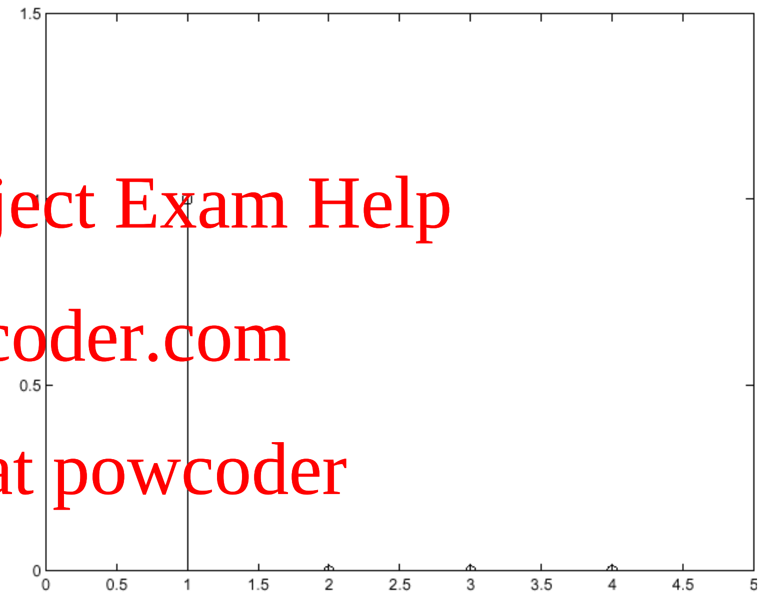
The entropy is characteristic of a given source of symbols

H is *highest* (equal to $\log_2 N$) if all symbols are equally probable

Entropy is *small* (always ≥ 0) when some symbols that are much more likely to appear than other symbols

ENTROPY EXAMPLE





$\{0.25\ 0.25\ 0.25\ 0.25\}$

$$H = -4\tfrac{1}{4}\log_2\left(\tfrac{1}{4}\right) = 2$$

$\{1,0,0,0\}$

$$H = 0$$

TAXONOMY OF COMPRESSION TECHNIQUES

Compression techniques are broadly classified into

- **Lossless Compression Techniques, also known as Entropy Coding**
- **Lossy Compression Techniques**

Refer to lecture for a description of taxonomy

EXAMPLES OF ENTROPY ENCODING

Types of techniques vary depending on how much you statistically analyze the signal

Run-length Encoding

Sequence of elements c_1, c_i… is mapped to a run – (c_i, l_i) where c_i = code and l_i = length of the i^{th} run

Repetition Suppression

Repetitive occurrences of a specific character are replaced by a special flag ab000000000 $\Rightarrow$ abψ9

Pattern Substitution

A pattern, which occurs frequently, is replaced by a specific symbol eg *LZW*

Huffman Coding

Arithmetic Coding

LZW – PATTERN SUBSTITUTION

Algorithm :

1. Initialize the dictionary to contain all blocks of length one (D={a,b} in the example below).
2. Search for the longest block W which has appeared in the dictionary.
3. Encode W by its index in the dictionary.
4. Add W followed by the first symbol of the next block to the dictionary.
5. Go to Step 2.

PATTERN SUBSTITUTION – EXAMPLE

Data: a b b a a b b a a b a b b a a a a b a a b b a

0 1 1 0 2 4 2 6 5 5 7 3 0

Dictionary			
Index	Entry	Index	Entry
0	a	7	b a a
1	b	8	a b a
2	a b	9	a b b a
3	b b	10	a a a
4	b a	11	a a b
5	a a	12	b a a b
6	a b b	13	b b a

HUFFMAN CODING

Huffman code assignment procedure is based on the frequency of occurrence of a symbol. It uses a *binary tree structure* and the algorithm is as follows

- **The leaves of the binary tree are associated with the list of probabilities**
- **Take the two smallest probabilities in the list and make the corresponding nodes siblings. Generate an intermediate node as their parent and label the branch from parent to one of the child nodes 1 and the branch from parent to the other child 0.**
- **Replace the probabilities and associated nodes in the list by the single new intermediate node with the sum of the two probabilities. If the list contains only one element, quit. Otherwise, go to step 2.**

HUFFMAN CODING (2)

Codeword formation:

Follow the path from the root of the tree to the symbol, and accumulates the labels of all the branches.

Example

***N* = 8 symbols: {a, b, c, d, e, f, g, h}**

3 bits per symbol (*N* =2^3 =8)

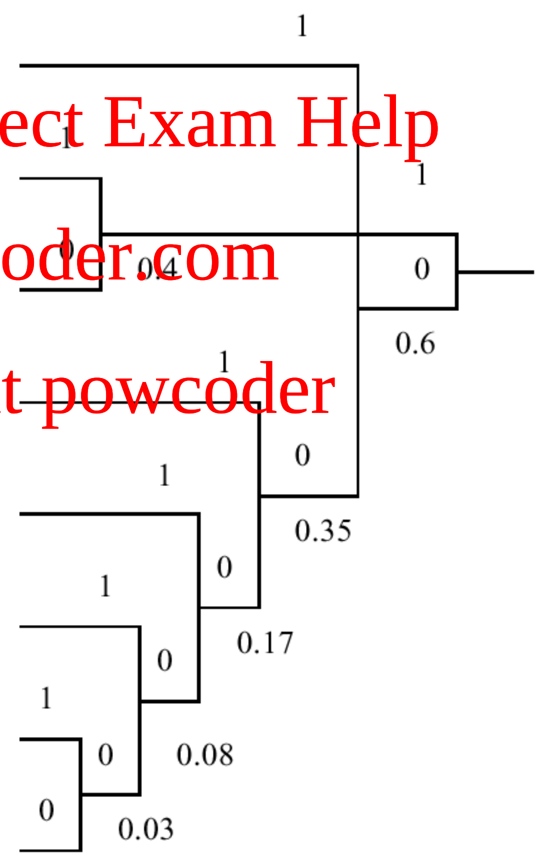
P(a) = 0.01, P(b)=0.02, P(c)=0.05, P(d)=0.09,
P(e)=0.18, P(f)=0.2, P(g)=0.2, P(h)=0.25

Average length per symbol $\underline{L}$ = $\sum_{i=1}^{i=8} 3P(i) = 3$

Entropy H = $H = -\sum_{i=1}^{i=8} P(i)\log_2 P(i) = 2.5828$ **= 0.86$\underline{L}$**

HUFFMAN CODING (2)

Original codewords	Huffman codewords	
111	01	P(h)=0.25
110	11	P(g)=0.2
101	10	P(f)=0.2
100	001	P(e)=0.18
011	0001	P(d)=0.09
010	00001	P(c)=0.05
001	000001	P(b)=0.02
000	000000	P(a)=0.01



HUFFMAN CODING (3)

Average Length per Symbol (with Huffman Coding)

$$\bar{L}_{Huff} = \sum_{i=1}^{i=8} P(i)\,l(i) = 6x0.01 + 6x0.02 + 5x0.05 + 4x0.09 + 3x0.18 + 2x0.2 + 2x0.2 + 2x0.25 = 2.63$$

Efficiency of the Encoder = H/ $\bar{L}_{Huff}$ = 98%

CCITT GROUP 3 1-D FAX ENCODING

Facsimile: *bilevel* image, 8.5” x 11” page scanned (left-to-right) at 200 dpi: 3.74 Mb = (1728 pixels on each line)

Encoding Process

- Count each run of white/black pixels
- Encode the run-lengths with Huffman coding

Probabilities of run-lengths are not the same for black and white pixels

Decompose each run-length n as $n = k \times 64 + m$ (with $0 \leq m < 64$), and create Huffman tables for both k and m

Special Codewords for end-of-line, end-of-page, synchronization

Average compression ratio on text documents: 20-to-1

ARITHMETIC CODING

Map a message of symbols to a real number interval. If you have a vocabulary of M symbols -

1. Divide the interval [0,1) into segments corresponding to the M symbols; the segment of each symbol has a length proportional to its probability.

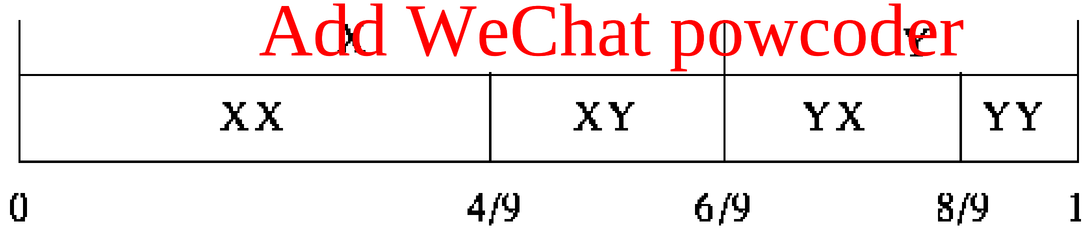
2. Choose the segment of the first symbol in the string message.

3. Divide the segment of this symbol again into M new segments with length proportional to the symbols probabilities.
4. From these new segments, choose the one corresponding to the next symbol in the message.
5. Continue steps 3) and 4) until the whole message is coded.
6. Represent the segment's value by a binary fraction.

ARITHMETIC CODING (2)

Suppose you have a vocabulary consisting of two symbols *X*, *Y* having prob(X) = 2/3 and prob(Y) = 1/3

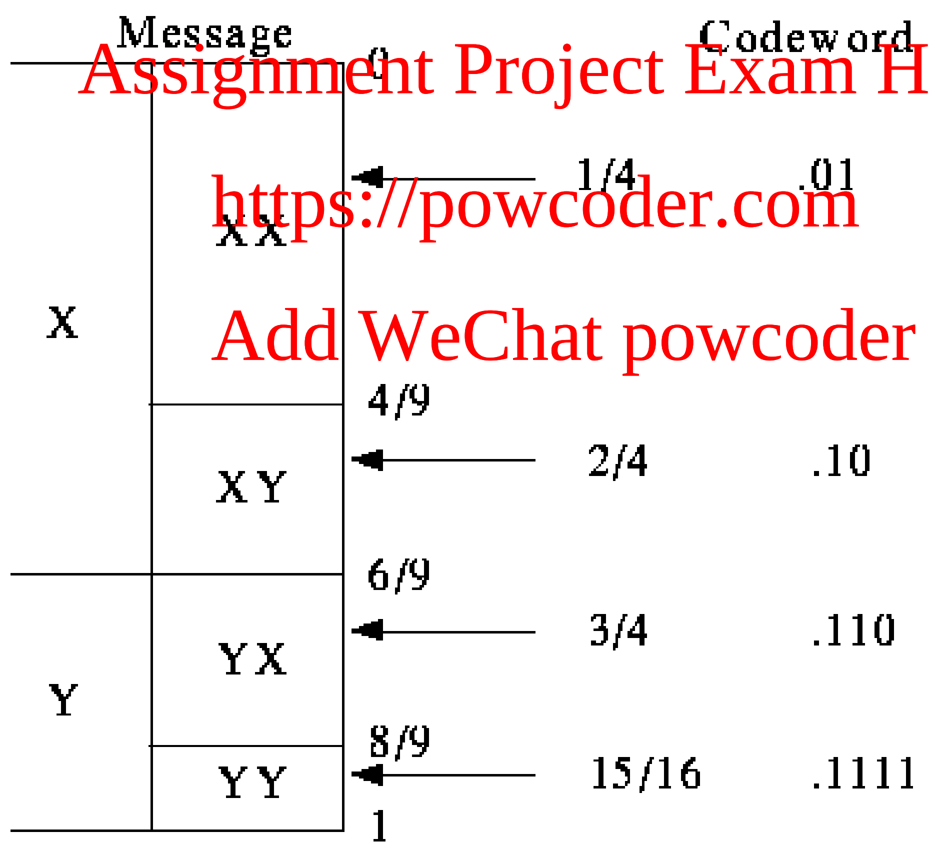
If we are only concerned with encoding length 2 messages, then we can map all possible messages to intervals in the range [0,1):



To encode message, just send enough bits of a binary fraction that uniquely specifies the interval.

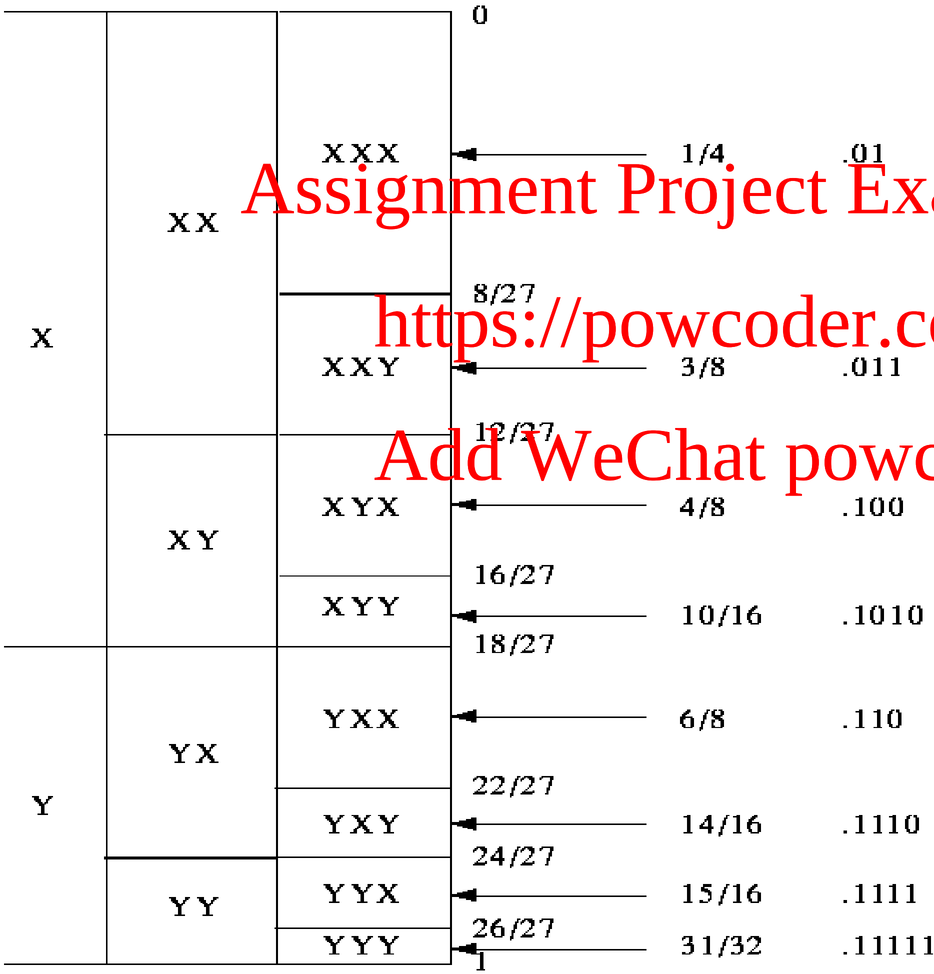
ARITHMETIC CODING (2)

Here is how code words are formed for the above example with message of length 2.



ARITHMETIC CODING (3)

Here is how code words are formed for the above example with message of length 3.



LOSSY COMPRESSION

Decompressed signal is not like original signal – data loss

Objective: minimize the *distortion* for a given compression ratio

- Ideally, we would optimize the system based on *perceptual distortion* (difficult to compute)
- We'll need a few more concepts from statistics…

STATISTICAL DEFINITIONS

y = the original value of a sample

$\hat{y}$ = the sample value after compression/decompression

$e = y - \hat{y}$ = error (or noise or distortion)

σ_y^2 = *power (variance)* of the signal

σ_e^2 = *power (variance)* of the error

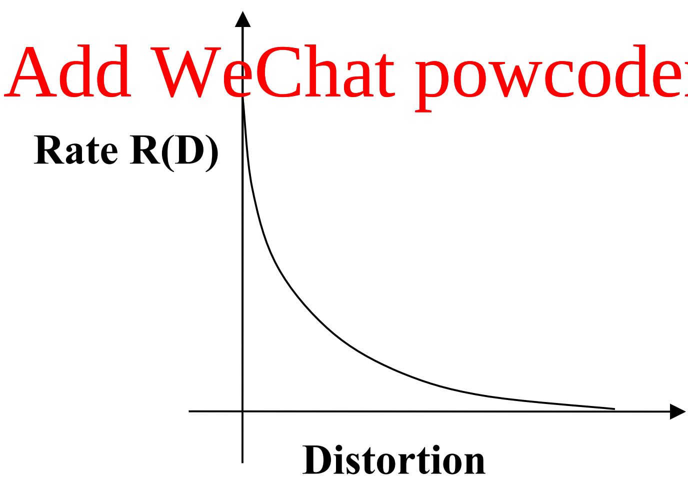
Signal to Noise Ratio (SNR) = $10 \log_{10} (\sigma_y^2 / \sigma_e^2)$

Peak Signal to Noise Ratio (PSNR) = $10 \log_{10} (\sigma_{peak}^2 / \sigma_e^2)$

RATE DISTORTION

Distortion measurement – difference between original and reconstructed signal.

Lossy Compression is always a tradeoff between rate (number of bits used) and distortion



LOSSY COMPRESSION: SIMPLE EXAMPLES

Subsampling: retain only samples on a *subsampling grid* (spatial/temporal). See examples in previous lecture

- **Compression achieved by reducing the number of samples**
- **Has fundamental limitations: see sampling theorem**

Quantization: quantize with fewer bits

- **Compression achieved by reducing the number of bits per sample**
- **As Quantization Interval size increases – compression increases (so does error!)**
- **Scalar Vs Vector Quantization**

Can we do better than simple quantization and subsampling?

DIFFERENTIAL PCM

If the *(n-1)*-th sample has value *y(n-1*), what is the "most probable" value for *y(n*)?

The simplest case: the *predicted* value for *y(n)* is *y(n-1)*

Differential PCM Encoding *(DPCM):*

- **Don't quantize and transmit *y(n)***

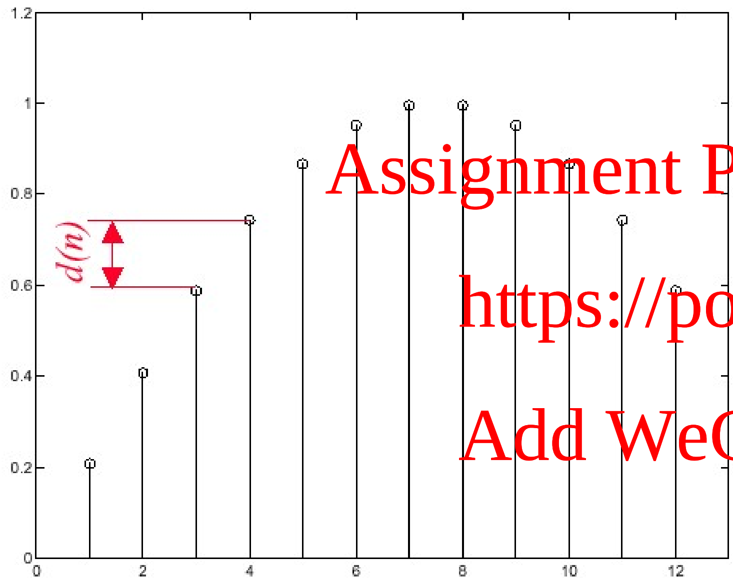
- **Quantize and transmit the residual *d(n)* = *y(n)-y(n-1)***

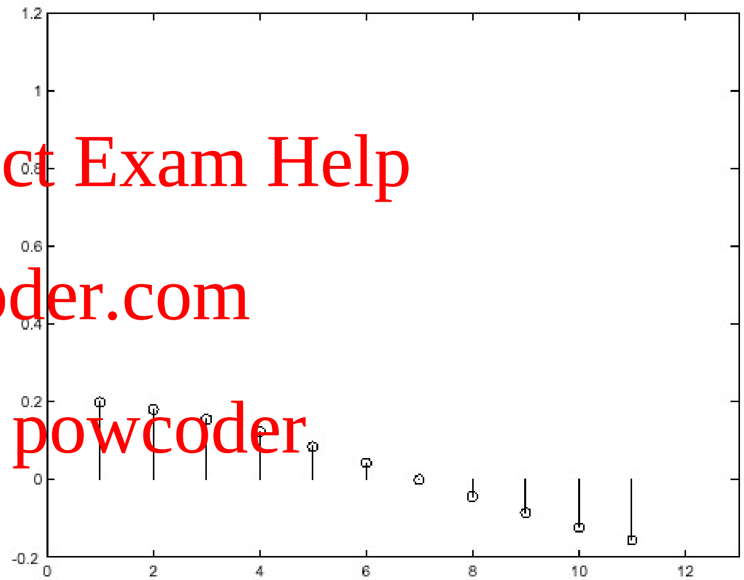
Decoding: $\hat{y}(n) = y(n-1) + \hat{d}(n)$

where $\hat{d}(n)$ **is the** quantized **version of *d(n)***

We can show that $\text{SNR}_{\text{DPCM}} > \text{SNR}_{\text{PCM}}$

DPCM – EXAMPLE





CLOSED-LOOP DPCM

In DPCM, the decoder reconstructs $\hat{y}(n) = y(n\text{-}1) + \hat{d}(n)$

The decoder must know $y(n\text{-}1)$!

In Closed Loop DPCM, instead of computing the difference

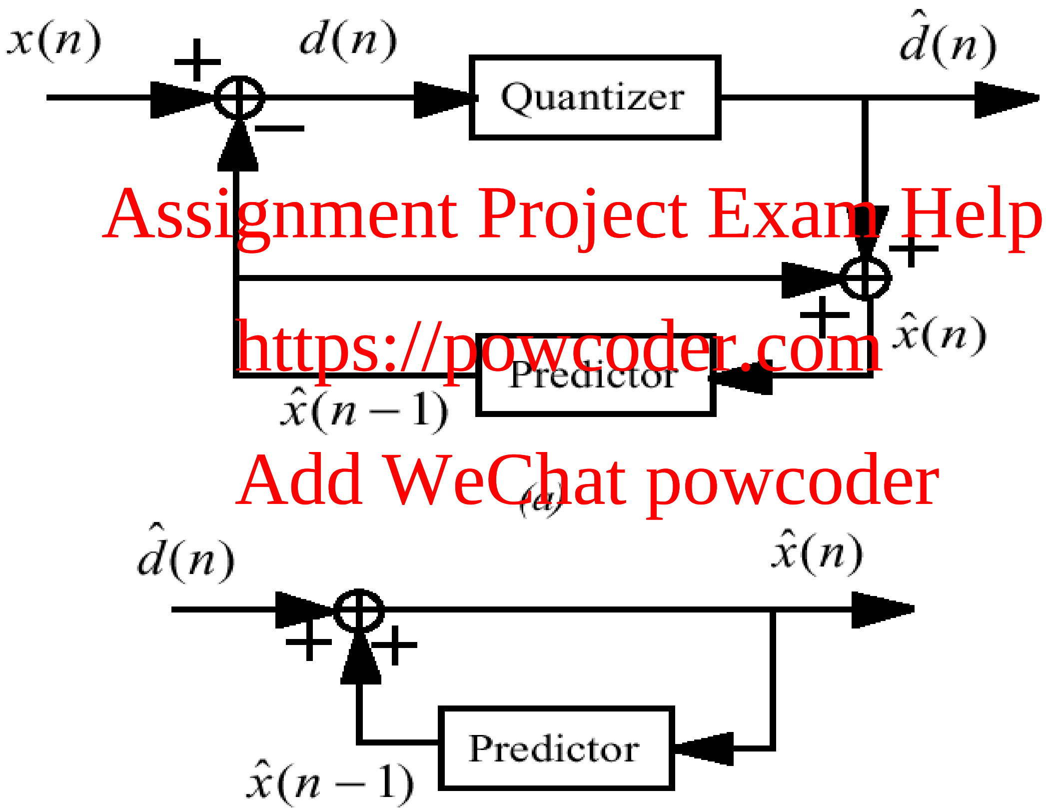
$d(n) = y(n) - y(n\text{-}1)$**, we compute**

$d(n) = y(n) - \hat{y}(n\text{-}1)$

Quantize $d(n)$ to get $\hat{d}(n)$ and transmit $\hat{d}(n)$

At the decoder, this implies $\Rightarrow \hat{y}(n) = \hat{y}(n\text{-}1) + \hat{d}(n)$

CLOSED-LOOP DPCM



(a)

(b)

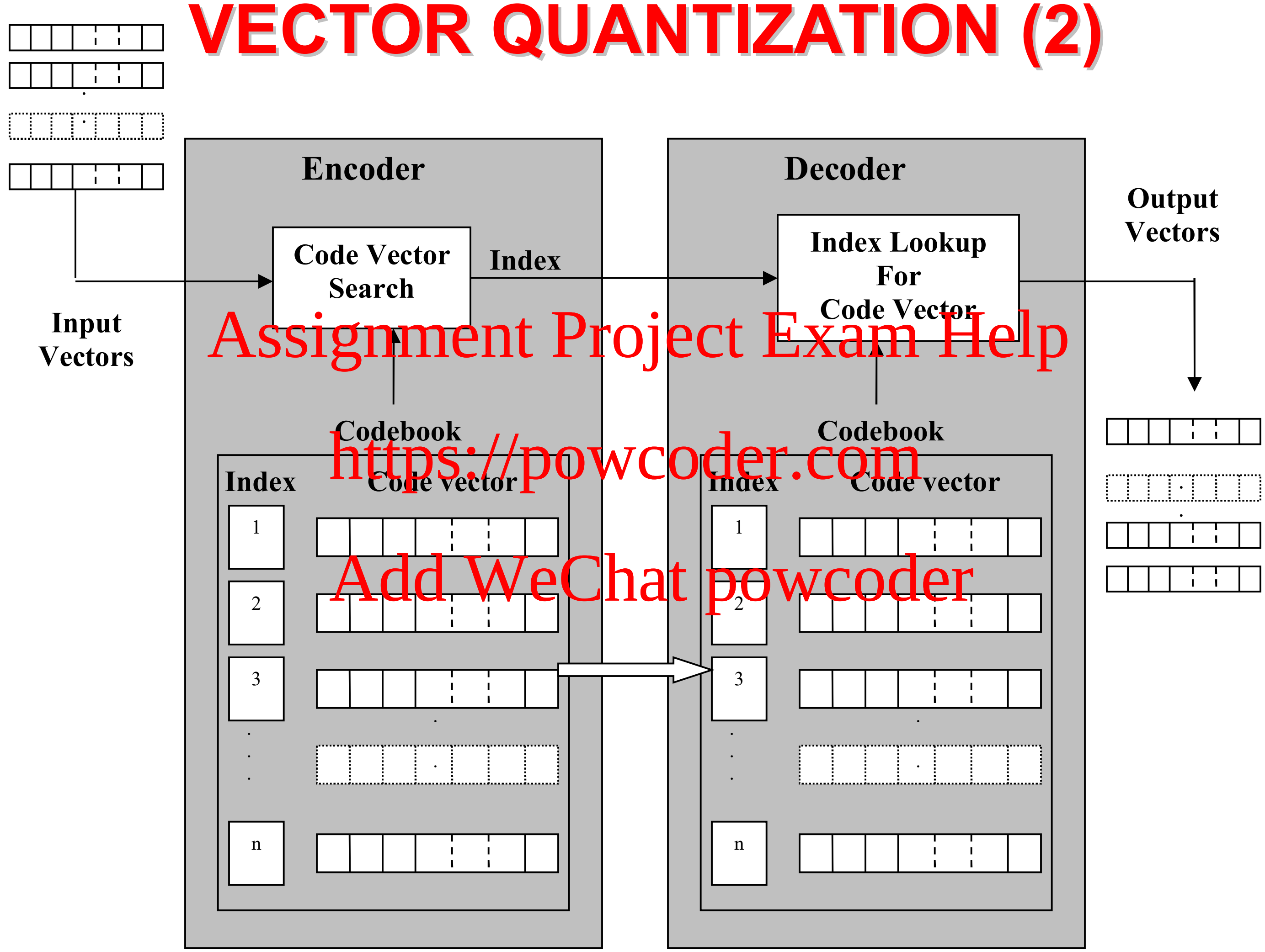
Closed-loop differential encoder (a) and decoder (b)

VECTOR QUANTIZATION

Vector Quantization works by

- Building a representative set of “code vectors” – this forms a codebook. Each code vector is represented by an index.
- Encoding works by partitioning the input signal into blocks or vectors. Each input signal block is then coded by an index which corresponds to the best matched block from the codebook
- Decoding works by a simple look up. Each encoded index can be replaced by its corresponding codebook block to reconstruct the original signal.

VECTOR QUANTIZATION (2)



TRANSFORM CODING

In Transform Coding a segment of information undergoes an invertible mathematical transformation.

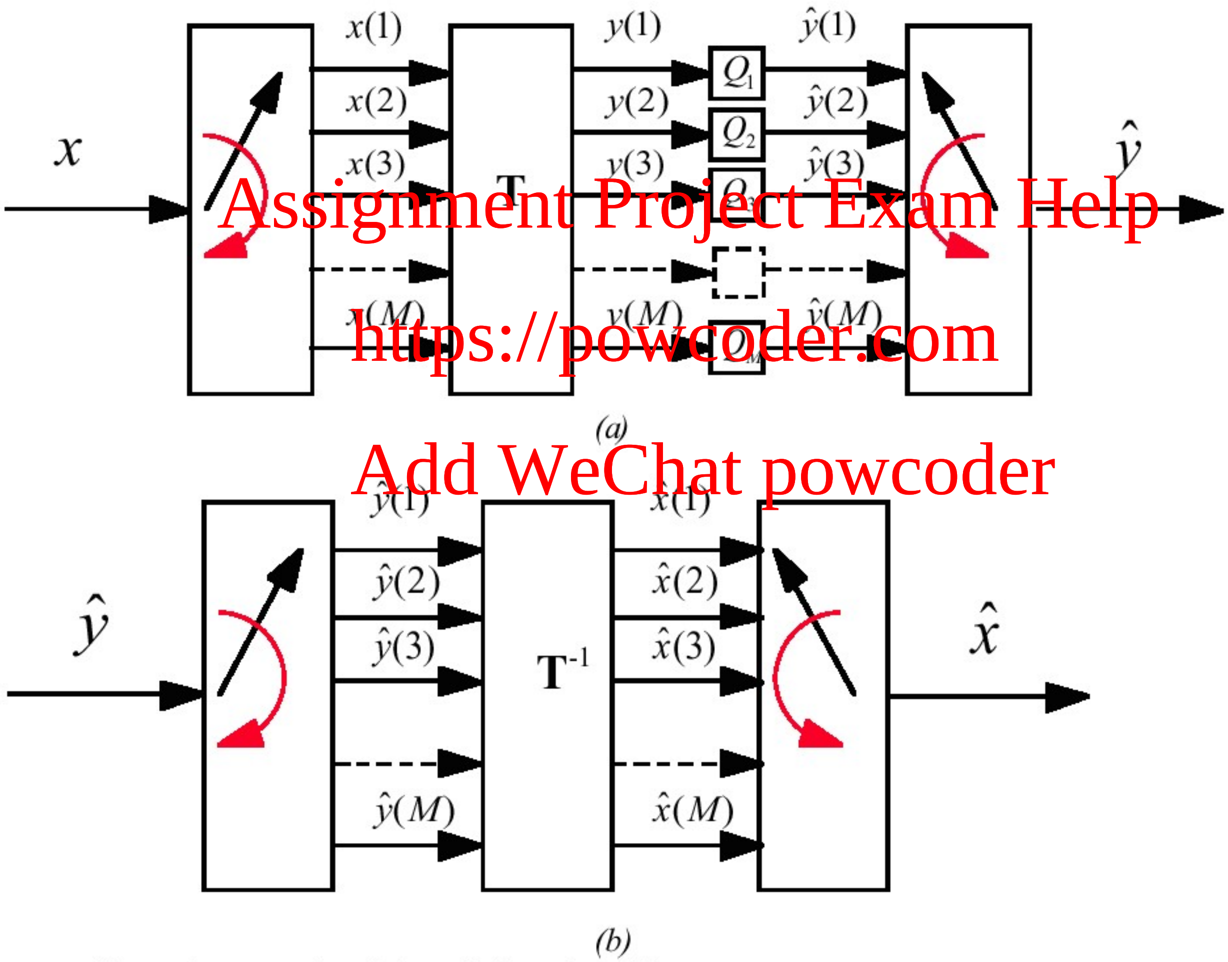
Segments of information can be samples (assume M samples) such as -

- **image frame**
- **8x8 pixel block (*M* =64)**
- **a segment of speech**

Transform Coding works as follows -

- **Apply a suitable invertible transformation (typically a matrix multiplication)**
  x(1), x(2), ... ,x(M) $\Rightarrow$ y(1), y(2), ... ,y(M) (channels)
- **Quantize and transmit *y(1),...,y(M)***

TRANSFORM CODING (2)



Transform coder (a) and decoder (b).

ISSUES WITH TRANSFORM CODING

Quantizers in different channels may have different numbers of levels $\Rightarrow$ each channel ultimately might yield a different number of bits.

Bit budget *B* : if we we quantize *y(1),...,y(M)* using *B(1),...,B(M)* bits respectively, then

$$B = \sum_{i=1}^{i=M} B(i)$$

Optimal bit allocation: allocate more bits to those channels that have the highest variance

It can be shown that *the transformation increases the quantization SNR*

Transformation also reduces the signal entropy!

DISCRETE COSINE TRANSFORM (DCT)

The *DCT* is a kind of transform coding with matrix *T* defined as

$$t_{pm} = \begin{cases} \sqrt{2/M}\cos\left(\dfrac{\pi}{M}p(m+1/2)\right) & p = 1,2,\ldots,M-1 \\ \sqrt{1/M}\cos\left(\dfrac{\pi}{M}p(m+1/2)\right) & p = 0 \end{cases}$$

It has the property that different channels represent the signal power along different (spatial or temporal) *frequencies* **similarly to the (discrete) Fourier transform**

SUBBAND CODING

It is a kind of transform coding (although operating on the whole signal)

It is not implemented **as a matrix multiplication but as a** *bank of filters* **followed by** decimation **(i.e., subsampling)**

Again, each channel represents the signal energy content along different frequencies. Bits can then be allocated to the coding of an individual subband signal depending upon the energy within the band

A SIMPLE SCHEME OF SUBBAND CODING