Module-4
Non Probability based Source Coding

Dr. Markkandan S

School of Electronics Engineering (SENSE)
Vellore Institute of Technology
Chennai

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

LEMPEL-ZIV ALGORITHM

Lempel-Ziv Algorithm

- Huffman coding requires symbol probabilities. But most real life scenarios do not provide the symbol probabilities in advance. Huffman coding is optimal for DMS source where the occurrence of one symbol does not alter the probabilities of the subsequent symbols.
- It might be more efficient to use the statistical inter-dependence of the letters in the alphabet along with their individual probabilities of occurrence.
- **Lempel-ziv algorithm** does not need the source statistics.
- It is variable -to-fixed length source coding algorithm and belongs to the class of universal source coding algorithm

Lempel-Ziv Algorithm

- The compression of an arbitrary sequence of bits is possible by coding a series of 0's and 1's as some previous such string (prefix string) plus one new bit
- The new string is formed by adding the new bit to the previously used prefix string becomes a potential prefix string for further strings
- These variable length blocks are called as **Phrases**. The phrases are listed in dictionary which stores the existing phrases and their locations.
- in encoding a new phrase, we specify the location of the existing phrase in the dictionary and append the new letter

Lempel-Ziv Algorithm : Example

Code the String **101011011010101011**

1. We will begin by parsing it in to comma seperated phrases that represent strings that can be represented by a previous string as a prefix, plus a bit
2. The first bit 1 has no predecessors, it has a null prefix string and the one extra bit itself.

   1,01011011010101011
3. The same for next bit 0 it doesnot have any prefix string

   1,0,1011011010101011
4. So far our dictionary contains the strings '1' and '0'. Next we encounter a 1, but it already exists in our dictionary. Hence we proceed further. The 10 is obviously combination of prefix 1 and 0 so we have

   1,0,10,11011010101011
5. Continue this way, we will be parsed the string as **1,0,10,11,01,101,010,1011**

Lempel-Ziv Algorithm : Example

- **Step-6:** Construct the Dictionary as follows using the strings parsed. Since we have 8 strings 3 bit bianry positions are used

String	Position Number	Position Number in Binary
1	1	001
0	2	010
10	3	011
11	4	100
01	5	101
101	6	110
010	7	111
1011	8	-

Lempel-Ziv Algorithm : Example

Step-7: Check for Prefix Availability

String	Position Number	Position Number in Binary	Prefix
1	1	001	No
0	2	010	No
10	3	011	1
11	4	100	1
01	5	101	0
101	6	110	10
010	7	111	01
1011	8	-	101

Lempel-Ziv Algorithm : Example

Step-8: Identify the Position Number of Prefix

String	Position No.	Position No. in Binary	Prefix	Position No. of prefix
1	1	001	No	000
0	2	010	No	000
10	3	011	1	001
11	4	100	1	001
01	5	101	0	010
101	6	110	10	011
010	7	111	01	101
1011	8	-	101	110

Lempel-Ziv Algorithm : Example

Step-9: Write the Codeword such a way that, Position Number of that prefix with last bit of the string we considered

String	Position No.	Position No. in Binary	Prefix	Position No. of prefix	Code word
1	1	001	No	000	0001
0	2	010	No	000	0000
10	3	011	1	001	0010
11	4	100	1	001	0011
01	5	101	0	010	0101
101	6	110	10	011	0111
010	7	111	01	101	1010
1011	8	-	101	110	1101

Lempel-Ziv Algorithm : Example

- Hence for string, **101011011010101011**.
- The coded word is **000100000010001101010111101011101**.
- This code is not efficient, since codeword length is higher than given code.
- But the dictionary size will increase, if a large amount of data to be coded and eventually less number of codes will be used.
- Lempel-Ziv algorithm is useful for large files

Lempel-Ziv Algorithm : Example with strings

Code the String $THIS\_IS\_HIS\_HIT$

1. We will begin by parsing it in to comma seperated phrases that represent strings that can be represented by a previous string as a prefix, plus a character
2. The first letter **T** has no predecessors, it has a null prefix string.
   $T, HIS\_IS\_HIS\_HIT$
3. The same for next character **H** it doesnot have any prefix string
   $T, H, IS\_IS\_HIS\_HIT$
4. Keep parsing, eventually we will get $T, H, I, S, \_, IS, \_H, IS\_, HI, T$

Lempel-Ziv Algorithm : Example with strings

Step-5: Construct the Dictionary and codeword

String	Position No.	Prefix	Position No. of prefix	Code word
T	1	No	0	0T
H	2	No	0	0H
I	3	No	0	0I
S	4	No	0	0S
_	5	No	0	0_
IS	6	I	3	3S
_H	7	_	5	5H
*IS_*	8	IS	6	6_
HI	9	H	2	2I
T	10	T	1	1T

Lempel-Ziv Algorithm : Example with strings

The coded String for $THIS\_IS\_HIS\_HIT$ is $0T0H0I0S0\_3S5H6\_2I1T$.

- Lempel-ziv algorithm is widely used in practice. The compress and uncompress utilities of the UNIX operating system use a modified version of this algorithm.
- The standard algorithms for compressing binary files use codewords of 12 bits and transmit 1 extra bit to indicate a new sequence.
- Using such code, Lempel-Ziv algorithm can compress transmissions of English text by about 55 percent.

RUN LENGTH ENCODING

RUN LENGTH ENCODING (RLE)

- Used to reduce the size of a repeating string of characters . This repeating string is called **run**
- RLE encodes a run of symbols in to two bytes , a count and a symbol
- RLE can compress any type of data regardless of its information content, but the content of data to be compressed affects the compression ratio.
- RLE cannot achieve high compression ratios compared to other compression methods, but it is easy to implement and is quick to execute. It is supported by most bit map file formats such as TIFF, JPG, BMP, PCX and FAX machines

RUN LENGTH ENCODING (RLE)

- RLE used for compression of images in the PCX format.
- PCX was the initial image format in DOS environment
- Now PCX is replaced by JPEG, BMP, PNG compression methods

EXAMPLE: RUN LENGTH ENCODING (RLE)

Consider the following bit stream

S=111111111111000000000000000000001110001100000.

1. This can be coded in to (12,1), (19,0),(3,1),(3,0), (2,1),(5,0)
2. Maximum repetition is 19, which can be coded in to 5 bits
3. The encoded bit stream is (01100,1), (10011,0), (00011,1), (00011,0), (00010,1), (00101,0)
4. Number of transmitted bits : 44
5. Number of encoded bits: 6 symbols X 6 = 36 bits
6. Compression Ratio is 36:44 = 1:1.22

RATE DISTORTION FUNCTION

Rate Distortion Function

- Although we live in an anlog world, most of the communication takes place in the digital form. Since most natural sources are analog, they are first sampled, quantized and then processed
- However, this representation of an arbitrary real number requires an infinite number of bits. Thus a finite representation of a continuous random variable can be never be perfect
- Consider an Analog message waveform x(t), which is a sample waveform of a stochastic process X(t). Assuming X(t) is a band limited, statioanry process, it can be represented by a sequence of non uniform samples taken at Nyquist rate.
- These samples are quantized in amplitude and encoded as a sequence of bianry digits.

Rate Distortion Function

A simple encoding strategy can be used to define L levels and encode every sample using

$$R = log_2 L \ bits, \ if \ L \ is \ a \ power \ of \ 2 \ or$$

$$R = \lfloor log_2 L \rfloor + 1 \ bits, \ if \ L \ is \ not \ a \ power \ of \ 2$$

The Squared Error Distrotion is defined as

$$d(x_k, \tilde{x_k}) = (x_k - \tilde{x_k})^2$$

In general distortion measure may be represented as

$$d(x_k, \tilde{x_k}) = |x_k - \tilde{x_k}|^p$$

Distortion and Rate Distortion Function

The Distrotion between a sequence of n samples $\mathbf{X_n}$, and their corresponding n quantised Values $\tilde{\mathbf{X}_n}$ is defined as

$$D = E[d(\mathbf{X_n}, \tilde{\mathbf{X}_n})] = \frac{1}{n}\sum_{k=1}^{n} E[d(x_k, \tilde{x_k})] = E[d(x_k, \tilde{x_k})]$$

The minimum rate (in bits/souurce output) required to represent the output X of the memoryless source with a distortion less than or equal to D is called the Rate Distortion Function R(D) is defined as

$$R(D) = \min_{p(\tilde{x}|x):E[d(\mathbf{X},\tilde{\mathbf{X}})]\leq D} I(\mathbf{X};\tilde{\mathbf{X}})$$

The distortion rate function for a discrete time, memoryless Gaussian source is defined as

$$D_g(R) = 2^{-2R}\sigma_x^2$$

TRANSFORM CODING

Image Compression Methods

- High quality images are represented by very large data sets
- Applications that involve imagery seem to be inherently linked to immediate human consumption, and so need to be fast in execution on computers and in transmission.
- Imagery has the quality of higher redundancy than we can generally expect in arbitrary data. For example, a pair of adjacent horizontal lines in an image are nearly identical (typically), while, two adjacent lines in a book have essentially no commonality.
- The human eye is very tolerant to approximation error in an image. Thus, it may be possible to compress the image data in a manner in which the less important information (to the human eye) can be dropped.

Image Compression Methods

- That is, by trading off some of the quality of the image we might obtain **lossy compression**, as opposed to the **lossless compression**
- Lossy compression can only be applied to data such as images and audio for which human beings will tolerate some loss of fidelity.
- That is, by trading off some of the quality of the image we might obtain lossy compression, as opposed to the lossless compression
- **JPEG** compression standard is actually a description of 29 distinct coding systems fro compression images

JPEG Standard for Lossless Compression

There are eight prediction methods available in the JPEG coding standards. One of the eight (which is the no prediction option) is not used for the lossless coding option that we are examining here. The other seven may be divided into the following categories:

- Predict the next pixel on the line as having the same value as the last one.
- Predict the next pixel on the line as having the same value as the pixel in this position on the previous line (that is, above it).
- Predict the next pixel on the line as having a value related to a combination of the previous, above and previous to the above pixel values. One such combination is simply the average of the other three.

JPEG Standard for Lossy Compression

- The JPEG standard includes a set of sophisticated lossy compression options which resulted from much experimentation by the creators of JPEG with regard to human acceptance of types of image distortion.
- The JPEG standard was the result of years of effort by the JPEG which was formed as a joint effort by two large, standing, standards organizations, the CCITT (The European telecommunications standards organization) and the ISO (International Standards Organization).
- The stages of lossy compression algorithm are
  - Lossy image Simplification - To Remove image complexity
  - Lossless compression step - Based on predictive filtering
  - Huffman or Arithmatic Coding

JPEG Standard for Lossy Compression - DCT

- The lossy image simplification step is based on Discrete Cosine Transform(DCT)

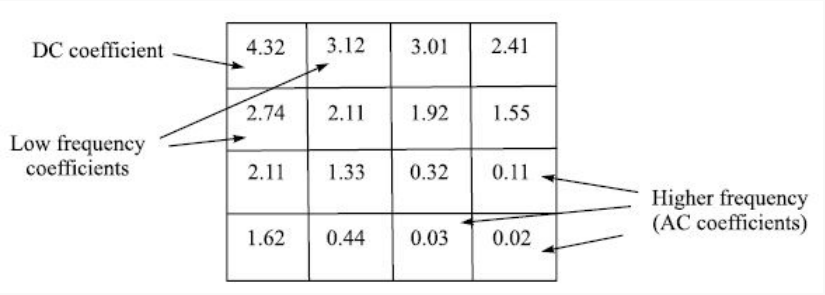
$$Y(k,l) = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} 4y(i,j) cos\left(\frac{\pi k}{2N}(2i+1)\right) cos\left(\frac{\pi l}{2M}(2j+1)\right)$$

  Where input image is NXM Pixels, y(i,j) is the intensity of the pixel in row i and column j.

- For most iamges, much of the signal energy lies at lower frequencies, which appear in the upper left corner of the DCT.
- The lower right values represent higher frequencies, often small
- DCT is computationally intensive with complexity of $O(N^2)$. Hence images are divided in to blocks

JPEG Standard for Lossy Compression - image Reduction

- DCT is applied to 8 by 8 pixel blocks of the image
- The 64 pixel values in each block are transformed by DCT into a new set of 64 values as DCT coefficients
- DCT coefficients represent spatial frequency of the image sub-block with AC and DC coefficients.

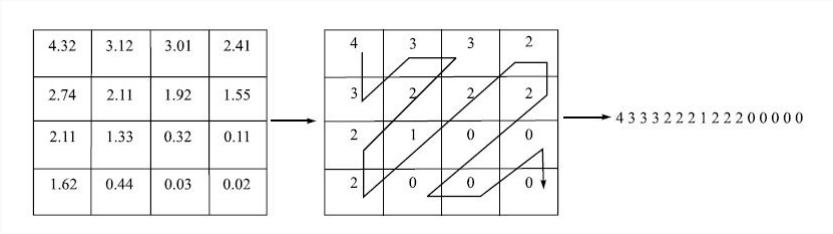


JPEG Standard for Lossy Compression - Image Reduction

- Due to nature of most natural iamges, maximum energy lies in low frequency as opposed to high frequency.
- For lossy compression following steps are followed
  1. First the lowest weights are trimmed by setting them to zero
  2. The remaining weights are quantized (that is, rounded off to the nearest of some number of discrete code represented values), some more coarsely than others according to observed levels of sensitivity of viewers to these degradations.
  3. Then several losssless compression methods are applied. DC coeeficients, vary slowly from one block to next block, Hence **predicion** is performed
  4. We have to send One DC coeeficient and difference between DC coefficients of surrounding blocks

JPEG Standard for Lossy Compression - Zig ZAg Coding

- The purpose of Zig-Zag coding is that we gradually move from the low frequency to high frequency, avoiding abrupt jumps in the values.
- Zig-Zag coding will lead to long runs of 0's, which are ideal for RLE followed by Huffman or Arithmetic Coding



Lossy Compression -JPEG

Image Compression Standards

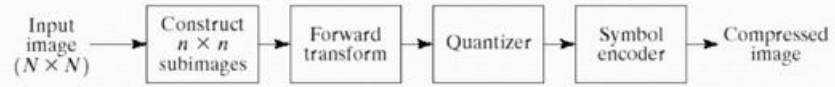
JPEG Encoding

Original

JPEG 27:1

Lossy Compression -Transform Coding

Lossy Compression
Transform Coding



a
b

FIGURE 8.28 A transform coding system: (a) encoder; (b) decoder.

The goal of the transformation process is to decorrelate the pixels of each sub-image, or to pack as much information as possible into the smallest number of transform coefficients

Lossy Compression
Transform Coding

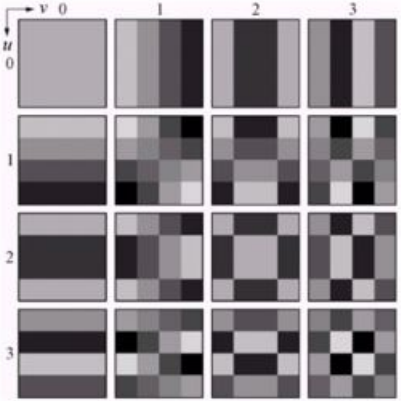


FIGURE 8.30 Discrete-cosine basis functions for $N = 4$. The origin of each block is at its top left.

Lossy Compression -Transform Coding

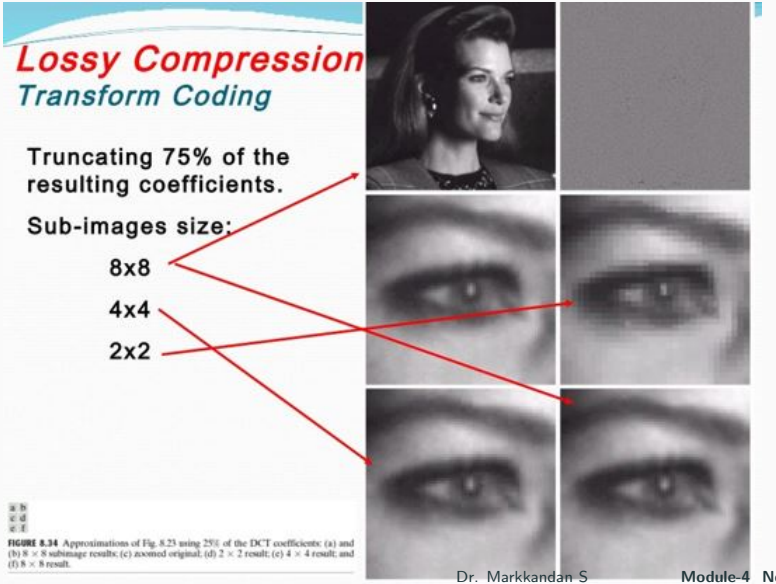
Lossy Compression
Transform Coding

1. Dividing the image into sub-images of size 8x8
2. Representing each sub-image using one of the transforms
3. Truncating 50% of the resulting coefficients
4. Taking the inverse Transform of the truncated coefficients

a b
c d
e f

FIGURE 8.31 Approximations of Fig. 8.23 using the (a) Fourier, (c) Hadamard, and (e) cosine transforms, together with the corresponding scaled error images.

Lossy Compression -Transform Coding



a b
c d
e f

FIGURE 8.34 Approximations of Fig. 8.23 using 25% of the DCT coefficients: (a) and (b) 8×8 subimage results; (c) zoomed original; (d) 2×2 result; (e) 4×4 result; and (f) 8×8 result.

Lossy Compression -Transform Coding