EEE116 Multimedia Systems

Topic 4 – Introduction to Data Compression

- Why Compression?
- Digital representation (Revisited)
  - Fixed length codes
- Variable length codes.
  - Information and entropy.
  - Shannon’s entropy measurement.
  - Huffman coding.
- Coding efficiency and redundancy.
- Data modelling for compression.

Dr. Charith Abhayaratne
c.abhayaratne@sheffield.ac.uk
F176 Mapping Building. Tel: Ext. 25893

Your needs:

1. Typing - say 20 characters/sec with 8 bits/character = 160 bits/sec
2. Speech conversation - 64K bits/sec
3. Music - ~200K bits/sec
4. Video – 144x176 pixels 15 fps – 256K bits/sec
   - 576x704 pixels 30 fps – 1-5M bits/sec

What you get (channel capacity values):

1. Standard modem - ~20K bits/sec
2. ADSL modem - ~1M bits/sec
3. Mobile phone - ~20K bits/sec
4. Computer terminal - ~10M bits/sec (peak)

Our goal is to reduce transmit time: Two options:

1. Increase the channel capacity (Problem set: Q3)
2. Reduce the data size (More feasible option for application users)

We have to reduce the data size, i.e., efficient digital representation leading to compression.

Class Activity -1

1. What are the units of data size?
2. Write down an expression for data size using the data rate
3. How can we reduce the data size?
4. Write down an expression for data rate using sampling frequency and the number of bits used per sample for digital representation?
5. Now discuss how to reduce the data rate.
6. What are your conclusions?

How to reduce data size?

- We can write the data size as a product of bit rate and time.

- One way is to reduce the time, i.e., the duration of data. This is not acceptable for multimedia content. You don't just want to listen to a part of a mp3 song file ☹

- Therefore, we should consider reducing the data bit rate.

- We know the data bit rate is a product of bits per sample and the sampling rate.

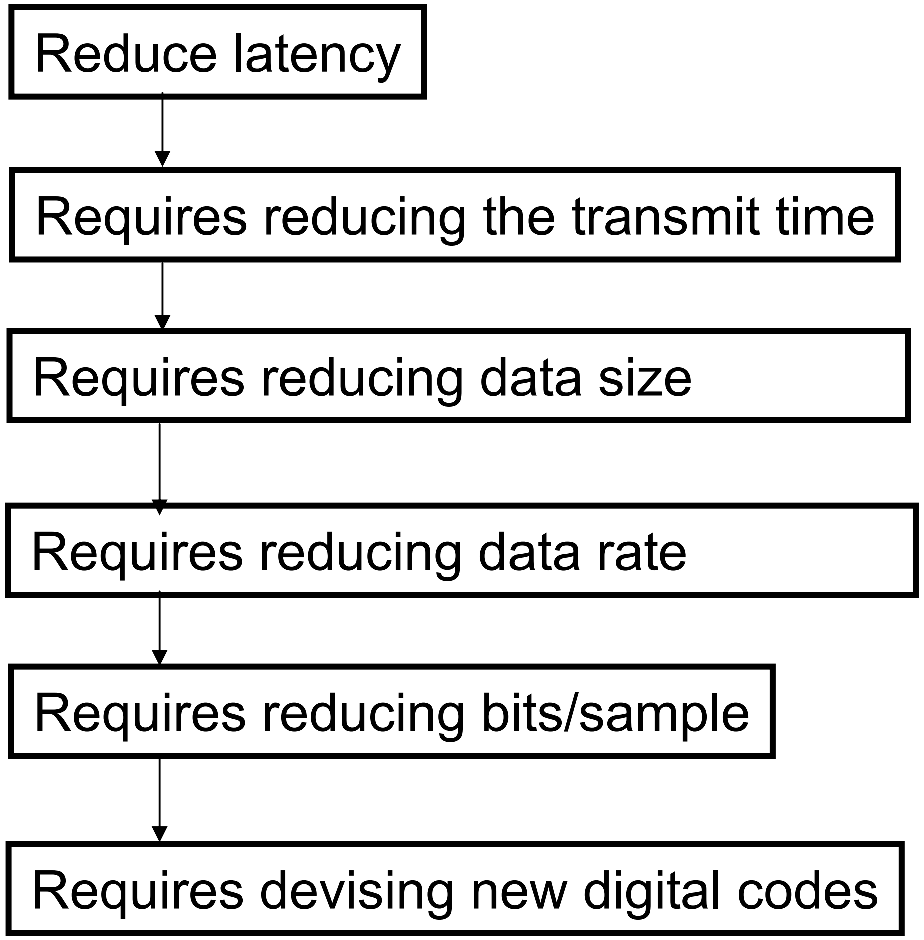
-We have two options here:

 -1. Reduce sampling frequency (Problem. What is the lowest limit of sampling?)

 -2. Reduce the code length, i.e., bits per sample.

The 2nd option is the most practical solution.

Road map to data compression


Reduce latency
Requires reducing the transmit time
Requires reducing data size
Requires reducing data rate
Requires reducing bits/sample
Requires devising new digital codes

So far we used $2^N=C$, to find an N bit code to represent C different values (symbols)

Can we do better? (The main learning outcome for the rest of this topic)

Class Activity -2

Consider the following sampled and measured values from a signal (-2, -1, 0,1,2) :

-2 0 1 1 0 2 -1 0 1 2 -1 -1 0 2 0 1 0 0 0 -1 0 1 1 1 0 -1 2 1 -1 0 0 2 0 -1 0
1 0 -1 0 1 -1 0 1 0 2 1 -1 0 -2 0

Sample value	-2	-1	0	1	2
Number of samples					

Devise a digital code to represent these sample values in digital form.

Sample value	-2	-1	0	1	2
Digital code					

We call this a FIXED length binary code

How many bits in total are required to encode this message?

What is the average code length (average bits per sample)?

Class Activity - 3

Now consider a "VARIABLE length code" as shown in the following table.

Sample value	-2	-1	0	1	2
Digital code	1110	110	0	10	1111

Sample value	-2	-1	0	1	2
Digital code	1110	110	0	10	1111
Length of codes (bits)					
Number of samples	2	10	20	12	6
Total number of bits					

How many bits in total are required to encode this message?

What is the average code length (average bits per sample)?

Average code length

T= Total number of samples
n_i= Number of bits used for the codeword of the ith symbol .
s_i= Number of samples of the symbol ith. (also called the frequency of occurrence)
C= Total number of different symbols.

The total number of bits (B) is given by $$B = \sum_{i=1}^{C} s_i n_i$$

The average code length (bits per sample) is computed by dividing the above by total number of symbols T

$$B / T = \frac{1}{T} \sum_{i=1}^{C} s_i n_i = \sum_{i=1}^{C} \frac{s_i}{T} n_i = \sum_{i=1}^{C} p_i n_i$$

p_i is the probability of occurrence a sample with symbol i.

Fixed length codes

Using formula $C=2^N$, we derived **FIXED LENGTH** binary (digital) codes. In the following example C=5. Therefore we need N=3 bits.

Symbol value	-2	-1	0	1	2
Digital code	000	110	101	111	001
n_i (bits)	3	3	3	3	3
s_i	2	10	20	12	6
p_i	2/50	10/50	20/50	12/50	6/50

Total bits required: $3x2 + 3x10 + 3x20 + 3x12 + 3x6$

$3 \times (2+10+20+12+6)$

$3 \times 50 \quad = n_i\, T$ bits

Average code length = $(n_i\, T)/T = n_i$ bits/sample

= N-bit fixed length code.

Variable length codes (VLC)

The probability of symbols are taken into consideration to construct **VARIABLE LENGTH codes** (VLC). We can construct an optimum (i.e., the most compact) code, if the most frequent (highest probable) symbol has the smallest number of bits for its code and the least frequent (the least probable) symbol has the greatest number of bits.

Symbol value	-2	-1	0	1	2
Digital code	1111	110	0	10	1110
n_i (bits)	4	3	1	2	4
s_i (from T4.Q8)	2	10	20	12	6
p_i	2/50	10/50	20/50	12/50	6/50

Average code length: the sum of products of n_i and p_i

$$= 4 \times 2/50 + 3\times10/50 + 1\times20/50 + 2\times12/50 + 4 \times 6/50$$

$$= 106/50 = 2.12 \text{ bits per sample.}$$

This is lower than using the fixed length codes.

How to decode?

For fixed length codes, we know the length of a code is N bits. So we can read groups of N bits and refer to the code table to decode them.

How about for variable length codes?

Canyoureadthisclearly?whatismissing?

When the codes are variable length each code needs to be separated by a space (or and end of code symbol) in order to read unambiguously.

That means it requires defining a new symbol to represent a “space”. This will add more bits. Therefore, not very practical.

Use of a “space” can be avoided if the unambiguous codes are used.

An unambiguous code is uniquely decodable and satisfies the prefix condition.

Prefix condition is “no codeword is the prefix of another code” (Problem sheet Q6)

How to decode?

For the variable length code example:

Symbol value	-2	-1	0	1	2
Digital code	1110	110	0	10	1111

Assume a bit stream of 1011110011011101100101011111110

Now use each of the codewords in the table and try to separate the codes which match the bits in the bit stream

1011110011011101100101011111110 (the code 10 matches)
10 11110011011101100101011111110 (the code 1111 matches)
10 1111 0011011101100101011111110 (the code 0 matches)
continuing the matching we can get
10 1111 0 0 110 1110 110 0 10 10 1111 110
Now we can decode
1 2 0 0 -1 -2 -1 0 1 1 2 -1.

You can now attempt

- Q5 and Q6 of the tutorial problem set.

How to encode?

We can consider the probability of symbols to construct **VARIABL LENGTH codes** (VLC).

Can construct an optimum (i.e., most compact) code,
if the most frequent (highest probable) symbol has the smallest number of bits for its code;
and the least frequent (the least probable) symbol has the greatest number of bits.

Can we give a meaning to data?

Is there a relationship between the probability and the meaning of data?

If so can we use this to derive our variable length codes?

Data (revisiting from Lecture 1)

- Representation of
  - facts,
  - concepts, or instructions
- in a formalized manner suitable for
  - communication,
  - interpretation, or processing by
    - humans or by automatic means.

- E.g., any representations such as characters or analogue quantities to which meaning could be assigned.

Information content

Information (revisiting from Lecture 1)

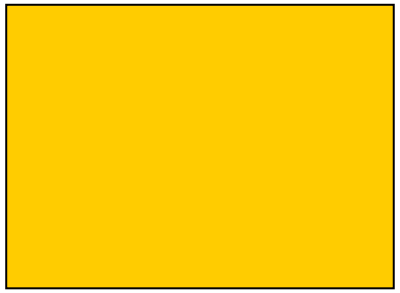
•The meaning that a human assigns to data

•by means of the known conventions used in their representation.

•"Entropy" is a measure of information content.

•Higher the entropy of a message, the more information it contains.

•The entropy is high if the uncertainty is high.



The same size of images. The data size is constant. But the amount of information we get varies.

How much information?

Message	How much Information?	What is the probability?
• Tomorrow is Saturday. (Saying this on a Friday)		
• It will rain tomorrow.		
• There will be no EEE116 lecture on next Friday. (Just joking) ☺		
• Next Saturday's lottery winning combination is 4, 8, 15, 16, 23, 42.		

Information content

From the previous slide we can conclude that the information content is inversely proportional to the probability of occurrence.

We measure the information content of an event (symbol) using entropy (H). We can interpret H as the minimum number of bits required to represent an event (symbol) in digital form.

$$I \propto \frac{1}{P} \qquad ----> 2^H = \frac{1}{P}$$

$$H = \log_2\left(\frac{1}{P}\right) \qquad \text{Bits}$$

$$= -\log_2(P)$$

Information content in the winning number combination

$= -\log_2(1/13{,}983{,}816)$

$= 23.7$ bits

Information content

The total entropy for a collection of symbols: The sum of the entropy of all symbols

$$H_{Total} = \sum_{i=1}^{C} s_i \log_2(1/p_i) \quad bits$$

C = Number of different groups of symbols
T = Total number of symbols
s_i = The frequency of occurrence of a symbol
p_i = The probability of occurrence of symbol
n_i = The number of bits used for the code of each symbol

The average entropy For C groups of different symbols in the message

$$H_{AVG} = \frac{H_{Tot}}{T} = \frac{1}{T}\sum_{i=1}^{C} s_i \log_2(1/p_i) = \sum_{i=1}^{C} \frac{s_i}{T} \log_2(1/p_i) = \boxed{\sum_{i=1}^{C} p_i \log_2(1/p_i)} \quad bits/sample$$

This is called Shannon's Entropy formula

Check the similarity between the Shannon's Entropy formula and the average code length formula:

$$Average \quad Code \quad Length = \boxed{\sum_{i=1}^{C} p_i n_i}$$

This shows, that by using the relationship, $n_i = log_2 (1/p_i)$, we can estimate the optimum number of bits per code of each symbol to device a VLC. Shannon's Entropy formula gives the theoretical minimum of the average code length that VLC can achieve.

In Fixed length codes we assumed the probability of C different symbols were equal. That means $p_i=1/C$. Therefore, we get $n_i = log_2 (1/p_i)= log_2 (C)$.

Class Activity-4 An Example

Symbol value	-2	-1	0	1	2
s_i	2	10	20	12	6
p_i	2/50	10/50	20/50	12/50	6/50
$-\log_2(p_i)$	4.64	2.32	1.32	2.06	3.06

$$H_{AVG} = \sum_{i=1}^{C} p_i \log_2(1/p_i)$$

H_{Avg} = sum of products of p_i and $-\log_2(p_i)$

= 2/50 x 4.64 + 10/50 x 2.32 + 20/50 x 1.32 + 12/50 x 2.06 + 6/50 x 3.06

= 2.04 bits/sample.

Compare with

N_{Avg} for fixed length codes 3 bits per sample. (Ref: Slide 9)

N_{Avg} for variable length codes 2.12 bits per sample. (Ref: Slide 10)

Coding efficiency and redundancy

Coding Efficiency (E) is the ratio of source entropy and the average code capacity, i.e., the average bits per symbol for the digital code.

E = (source entropy)/(average code length)

Redundancy (R) is a measure of how redundant the data are. Usually measured using the coding efficiency as follows:

R = 1-E

.

Coding efficiency and redundancy

Coding Efficiency (E)

E = (source entropy)/(average code length)

for fixed length codes E =

for variable length codes E =

Redundancy (R) is a measure of how redundant the data are.

R = 1-E

for fixed length codes R =

for variable length codes R =

Lossless coding

- In previous examples, we measured only the redundancy due to the digital representation of data.
- Therefore we call it coding redundancy.
- Variable length coding is also called entropy coding.
- Entropy coding methods can encode and decode symbols to recover the exact original data.
- This type of coding is called lossless coding.
- Huffman coding is an example for VLC and lossless coding.

Class activity - 5 : An Example

A Message consists of six symbols with following frequency of occurrence

Symbol	s_i	p_i	[illegible]
E	29		
I	5		
N	7		
P	12		
S	4		
T	8		

$\Sigma C_i = 65 \quad \Sigma\, p_i = 1.000$

Number of different symbols C = 6 A fixed length coder would require 3 bits/symbol.

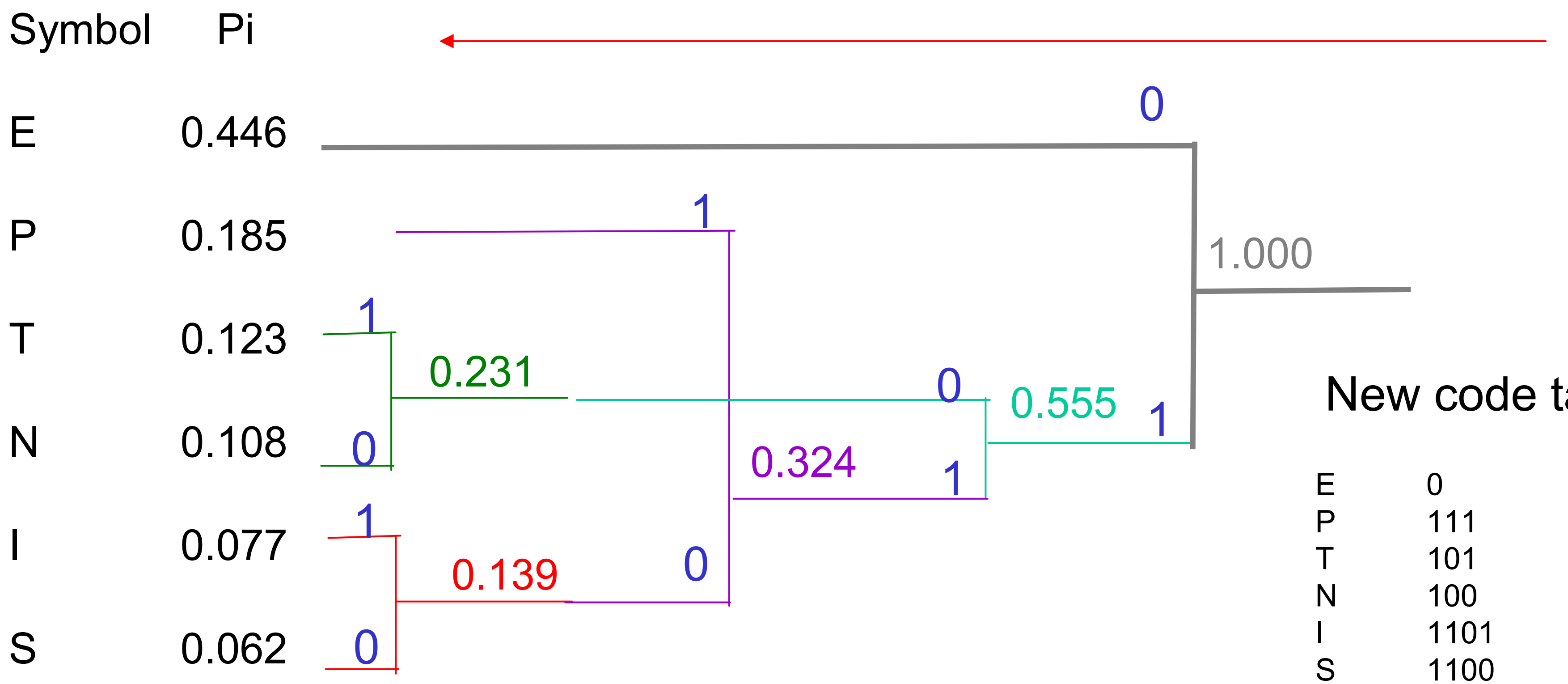
Entropy, H

or

Huffman Coding – As an example of VLC

Procedure:

1. List symbols in descending order of probability.
2. Group the two symbols of lowest probability, making a new symbol.
3. Assign digits 0/1 when a pair is merged.
4. Repeat 2 and 3 until symbols are exhausted.
5. Read the code from right to left.



New code table

E	0
P	111
T	101
N	100
I	1101
S	1100

Is this code unambiguous?

E	0
P	111
T	101
N	100
I	1101
S	1100

All that the receiver sees is a string of '0's and '1's

… 1101100110010101111 … What does this bit string spell?

INSTEP

Symbol	s_i	p_i	$-Log_2(p_i)$	n_i	n_ip_i
E	29	0.446	1.164	1	0.446
I	5	0.077	3.700	4	0.308
N	7	0.108	3.215	3	0.324
P	12	0.185	2.437	3	0.555
S	4	0.062	4.022	4	0.248
T	8	0.123	3.022	3	0.369
	$\Sigma = 65$	$\Sigma P_i = 1.000$	$-\Sigma n_i \log_2(p_i) = 2.22$		$\Sigma n_ip_i = 2.25$ bits/symbol

Coding efficiency E_C = (source entropy)/(code capacity)

For fixed length codes ? 2.22 / 3 = 74.0%

For Huffman codes ? 2.22 / 2.25 = 98.7%

- Huffman codes provide near optimum code sizes
- Limited by need to use integer number of bits per symbol. 100% efficient only when the probabilities are powers of (1/2).

  (Solution: Arithmetic coding)

- Can unambiguously read variable length code - prefix property
- Computational easy to encode and decode
- Both encoder and decoder need to use same code table

- Disadvantage - need the statistics of a message before sending it!
  - With dynamic or adaptive Huffman coding - encoder and decoder build up the code as the message is sent and use the same rules. Used in some modems and transmission of large text documents.
  - or use a predefined probability table.

Compression ratio

Compression ratio = original size / new size
Usually expressed as a ratio.

One of the following measures can be used to measures the data size:

Average code length
Total bits
Bit rate

For example: If data is originally represented using fixed length code and then encoded using the variable length in our previous example, what is the compression ratio?

$$\text{Compression ratio} = \frac{\text{(average bits per symbols in fixed length codes)}}{\text{(average bits per symbol using VLC)}}$$

$$= 3 / 2.12$$

$$= 1.42:1$$

You can now attempt

- Q7 of the tutorial problem set.

Entropy of different data sets

Symbol value	+	*	$	£	Totals
s_i	64	64	64	64	256
p_i					1
$-\log_2(p_i)$					

$H_{Avg}=$

Symbol value	+	*	$	£	Totals
s_i	192	32	24	8	256
p_i					1
$-\log_2(p_i)$					

$H_{Avg} =$

Symbol value	+	*	$	£	Totals
s_i	128	64	32	32	256
p_i					1
$-\log_2(p_i)$					

$H_{Avg} =$

Conclusion?

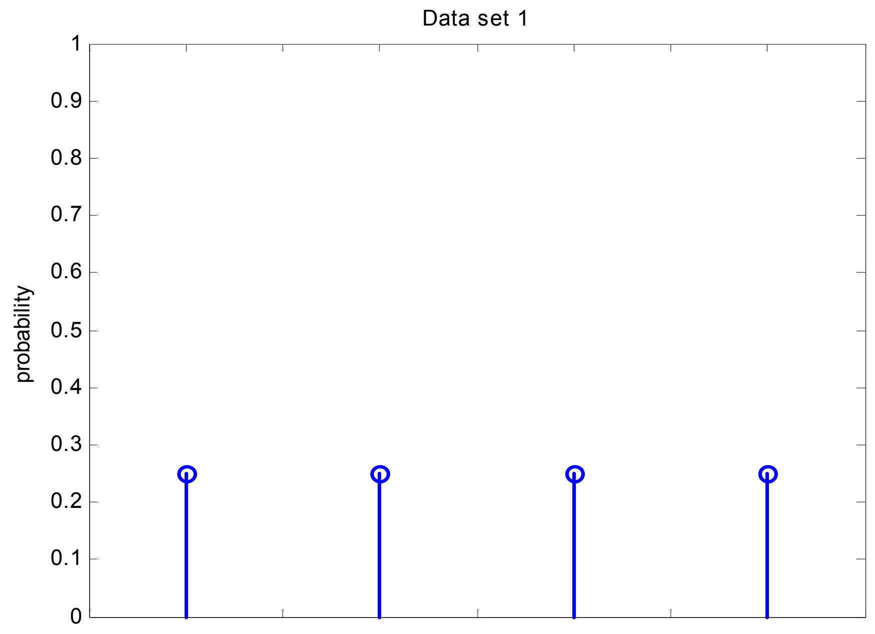
Entropy of different data sets

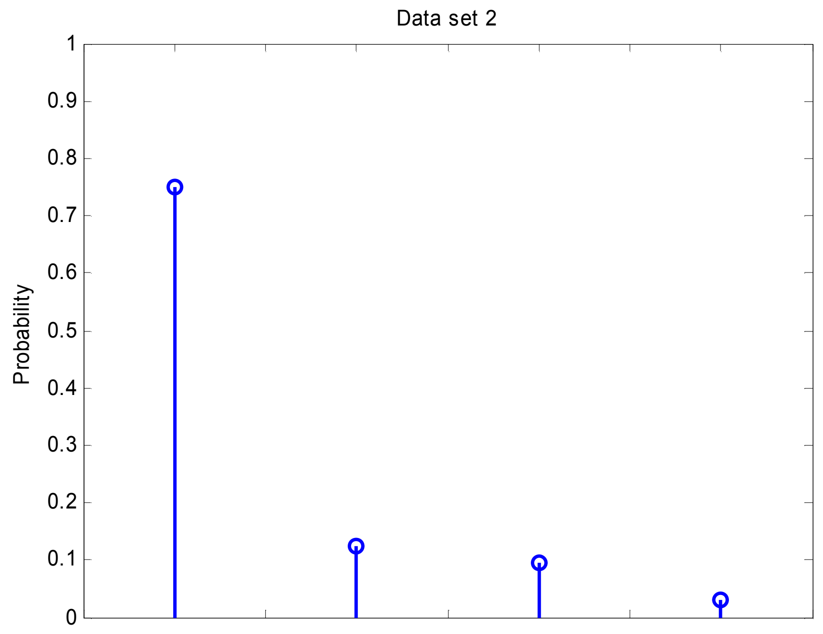
Good News!

From this week:

Friday's lecture starts at 11.10 am in SG-LT08

Entropy of different data sets





H_{Avg}= 2 bits per sample.

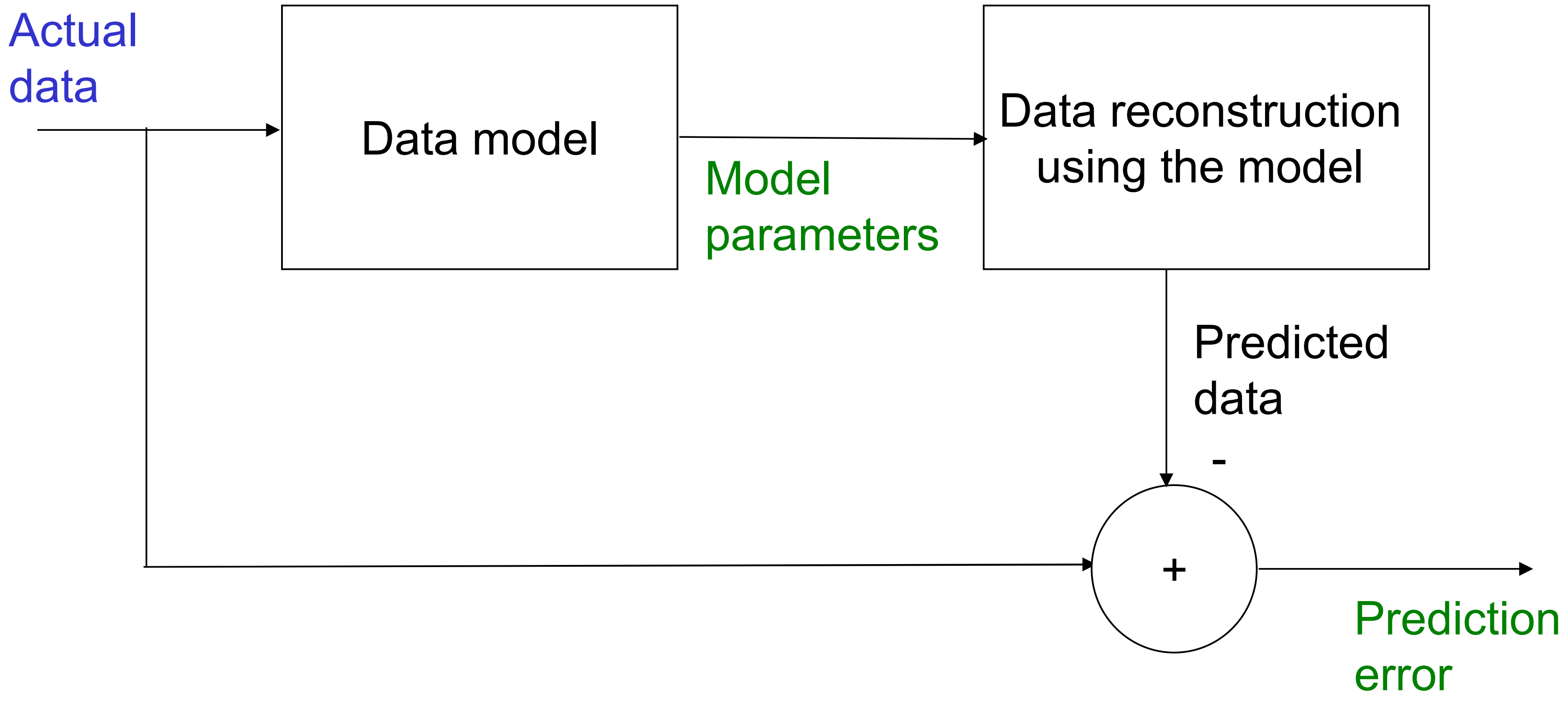
Probabilities are equal. High entropy

H_{Avg} = 1.17 bits per sample.

High peak and long tail in the distribution. Low entropy.

If we can re-arrange data to get a histogram like in right hand side, then we can use fewer bits to represent data. This can be done by modelling data to remove inter sample data redundancy. Also called decorrelation.

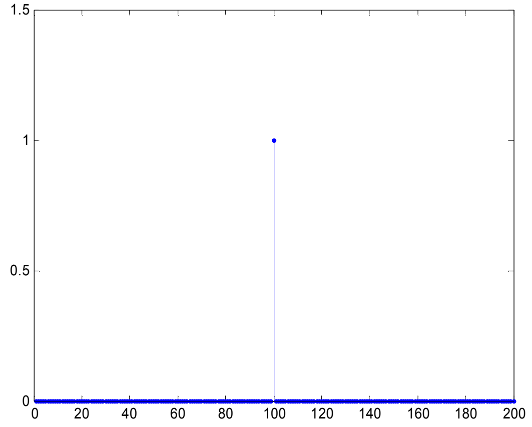
Data modelling



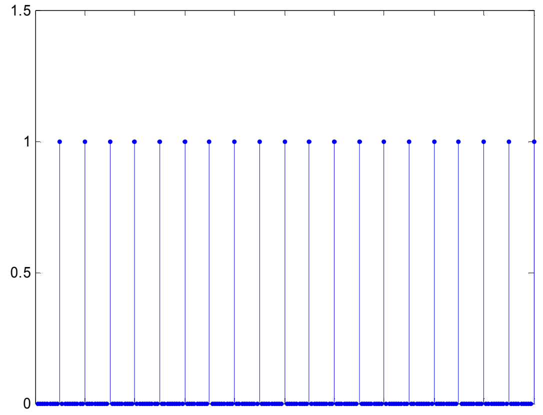
Now the actual data stream is replaced by the model parameters + prediction error

Sampling (revisited)

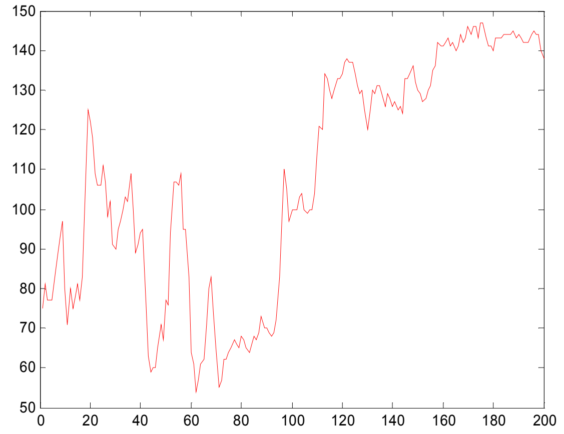
Unit impulse



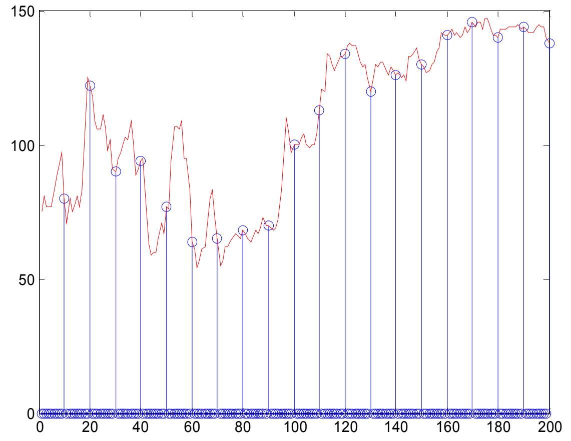
Unit pulse train (1)



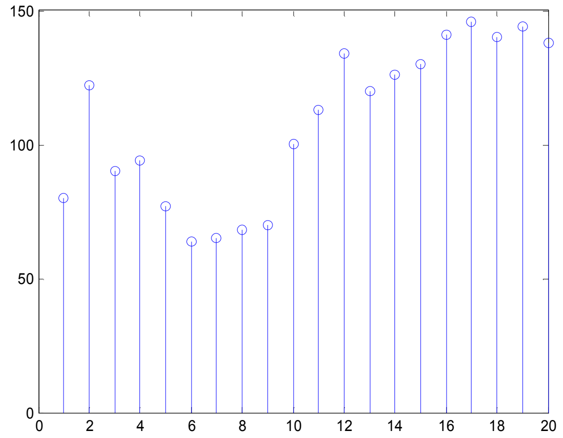
This is also called Pulse Coded Modulation (PCM)
i.e., coding the actual value (subject to quantisation)



Continuous signal *g(x)* *(2)*



Multiplication of (1) and (2)



The sampled signal $g_s(x)$

Data modelling

We have seen that the entropy varies according to the probability distribution.

If the distribution is flat high entropy and if it is skewed then low entropy.

We can try to fit data into a model in order to change the probability distribution.

If data is correlated we can easily predict the i^{th} symbol using $(i-1)^{th}$ symbol and record the prediction error instead of the symbol value.

We call this "Differential Pulse Coded Modulation" (DPCM) using an order-1 model, since we use only the nearest previous neighbouring sample to predict the current model.

After the prediction data is said to be decorrelated and should have low entropy, that means require a fewer number of bits compared to the original correlated data.

DPCM Example

Consider the data set:

25 25 23 23 24 24 24 24 25 25 26 26 26 26 23 23

Using DPCM the data set can be transformed to:

Original data set:

Symbol value	25	23	24	26
s_i				
p_i				

$H_{Avg}=$

Data set after DPCM:

Symbol value					
s_i					
p_i					
$-\log_2(p_i)$					

$H_{Avg}=$.

Data modelling

We used "Differential Pulse Coded Modulation" (DPCM) as simple data model. We used an order-1 model, since we use only the nearest previous neighbouring sample to predict the current model.

At the encoder:

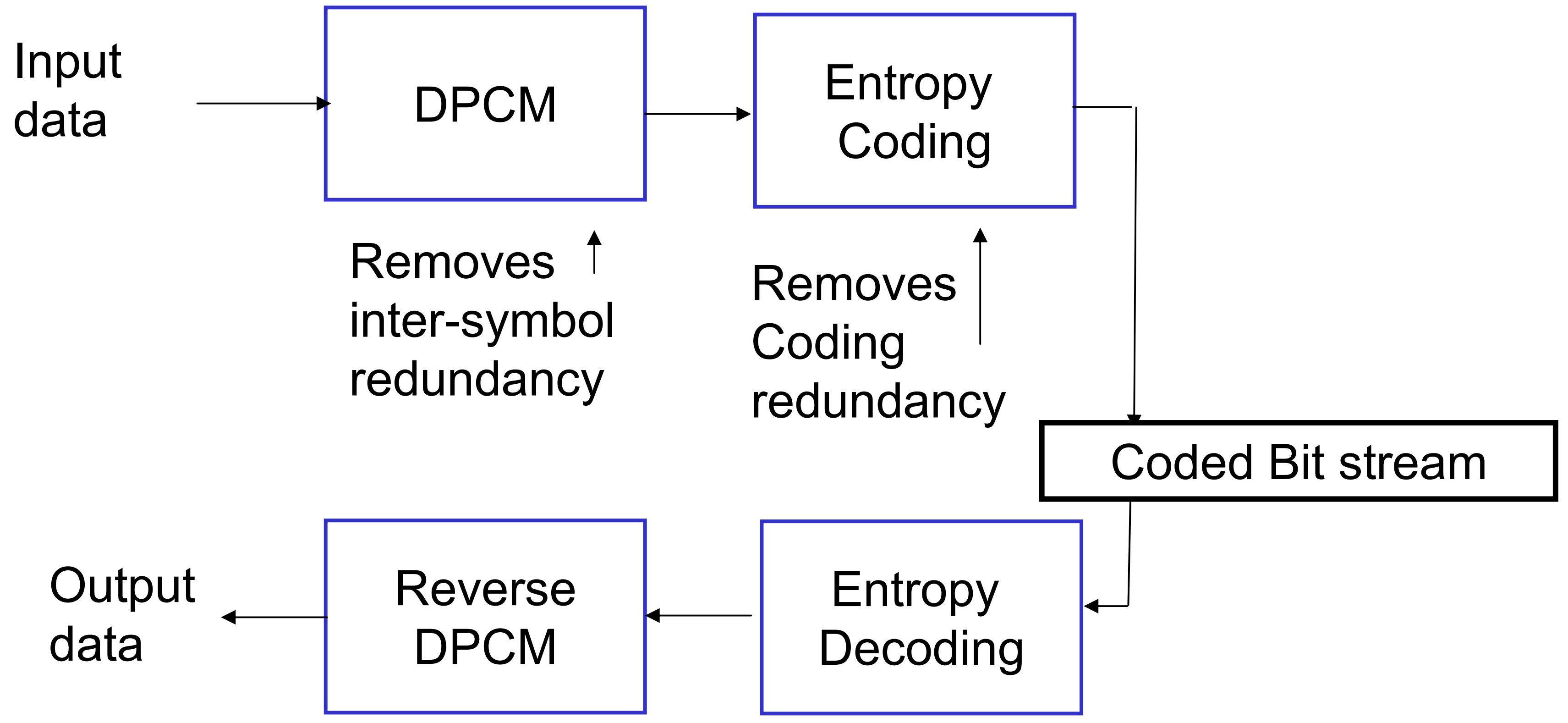
$$(\text{coded value})_t = (\text{sample value})_t - (\text{sample value})_{t-1}$$

At the decoder:

$$(\text{sample value})_t = (\text{coded value})_t + (\text{sample value})_{t-1}$$

To represent sample value at t, we only use the samples recorded earlier than t. We call such a scheme as a "causal" prediction.

Data Compression Model (a basic)



This results in a lossless compression scheme.

Can we do even better?
Yes, consider each types of data and model the limitations of human perception

You can now attempt up to

- Q8 of the tutorial problem set.

Note: Friday's lecture - new time @ 11.10 am
- new venue SG-LT08

Mid Term Test - Friday 27th March @ 11.10 am
in first year labs
-up to the end of Topic 5 (lecture on 20th March)
-up to Q10 of problem sheets
- SAQs in MOLE for topics 2, 3, 4 & 5.