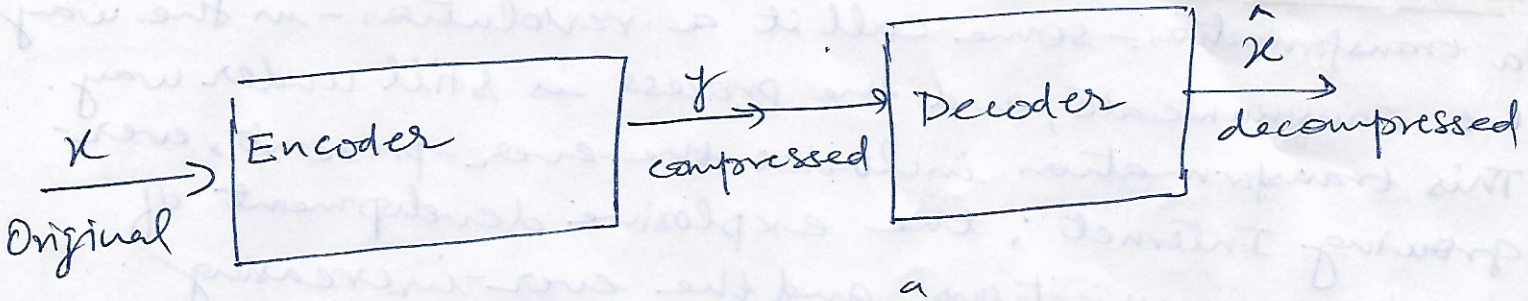
UNIT-1 Lecture-1

Introduction:- In the last decade we have been witnessing a transformation - some call it a revolution - in the way we communicate, and the process is still under way. This transformation includes the ever-present, ever-growing Internet; the explosive development of mobile communications; and the ever-increasing importance of video communication. Data compression is one of the enabling technologies for each of these aspects of the multimedia revolution. It would not be practical to put images, let alone audio and video, on websites if it were not for data compression algorithms. cellular phones would not be able to provide communication with increasing clarity were it not for compression. The advent of digital TV would not be possible without compression. Data compression, which for a long time was the domain of a relatively small group of engineers and scientists, is now ubiquitous (सर्वव्यापी). Make a long distance call and you are using compression. Use of modem or your fax machine, and you will benefit from compression. Listen to music or your mp3 player or watch a DVD and you are being entertained courtesy of compression.

Data compression algorithms are used to reduce the number of bits required to represent an image or a video sequence or music. It is a art of science of representing ~~import~~ information in a compact form. We create these compact representations by identifying and using structures that exists in the data. Data can be characters in a text file, numbers that are samples of speech or image waveforms or sequence of numbers that are generated by other processes.



- Lossless compression $x = \hat{x}$
  - Also called entropy coding, reversible coding.
- Lossy compression $x \neq \hat{x}$
  - Also called irreversible coding.
- Compression ratio = $|x| / |y|$
  - $|x|$ is number of bits in x.

Why compression

- Reduce storage space
- Reduce time for transmission
  - faster to encode, send then decode than to send the original
- Progressive transmission
  - some compression techniques allow us to send the most important bits first so we can get a low resolution version of some data before getting the high fidelity version.
- Reduce computation
  - Use less data to achieve an approximate answer

Compression techniques

There is a compression algorithm that takes an input x and generates a representation x that requires fewer bits, and there is a reconstruction ~~of these operations~~ algorithm that operates on the compressed representation x_c to generate the reconstruction y. These operations are shown in fig. We will follow convention and refer to both the compression and reconstruction algorithms together to mean the compression algorithm.

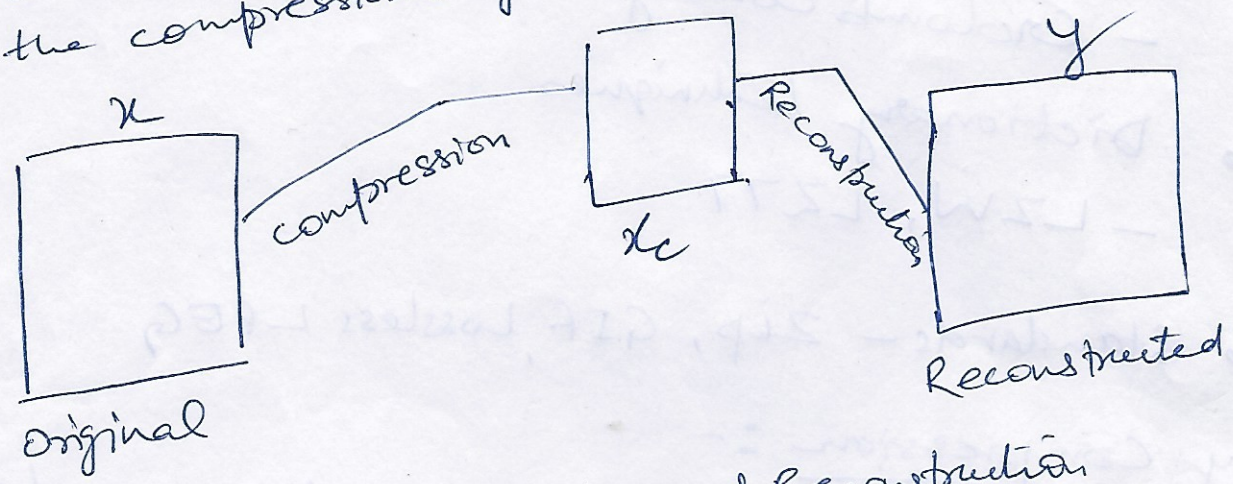


fig: compression and Reconstruction

Based on the requirements of reconstruction, data compression schemes can be divided into two broad classes; lossless schemes, in which y is identical to x, and lossy compression schemes which generally provide much higher compression than lossless compression but allows to y to be different from x.

Lossless compression

Lossless techniques as their name implies, involve no loss of information. If data have been losslessly compressed, the original data can be recovered exactly from the compressed data. Lossless compression is generally used for applications that cannot tolerate any difference between the the original and reconstructed data.

- Data is not lost - the original is really needed
  - Text compression
  - Compression of computer binary files

- Statistical Techniques
  - Huffman Coding
  - Arithmatic Coding
  - Golomb Coding
- Dictionary techniques
  - LZW, LZ77
- Standards - Zip, GIF, Lossless LPEG

Lossy Compression :-

Lossy compression techniques involve some loss of information, and data that have been compressed using lossy techniques generally cannot be recovered or reconstructed exactly. In return for accepting this distortion in the reconstruction, we can generally obtain much higher compression ratios than is possible with lossless compression.

In many applications, this lack of exact reconstruction is not a problem. For example, when storing or transmitting speech, the exact value of each sample of speech is not necessary. Depending on the quality required of the reconstructed speech, varying amounts of loss of information about the value of each sample can be tolerated.

- Data is lost, but not too much
  - audio
  - video
  - still images, medical images, photographs
- Major techniques include
  - Vector quantization
  - Wavelets
  - Block transforms
  - Standards – JPEG, JPEG 2000, MPEG 2

Why is data compression possible

- Most data from nature has redundancy
  - There is more data than the actual information contained in the data.
  - Squeezing out the excess data amounts to compression.
  - However, unsqueezing is necessary to be able to figure out what the data means.
- Information theory is needed to understand the limits of compression and give clues on how to compress well.

What is information

- Analog data
  - Also called continuous data
  - Represented by real numbers (or complex numbers)
- Digital data
  - Finite set of symbols $\{a_1, a_2 \cdots a_m\}$
  - All data represented as sequence in the symbol set.

Measures of Performance Lecture-3.

A compression algorithm can be evaluated in a number of different ways. We could measure the relative complexity of the algorithm, the memory required to implement the algorithm, how fast the algorithm performs on a given machine, the amount of compression and how closely the reconstruction resembles the original.

A very logical way of measuring how well a compression algorithm compresses a given set of data is to look at the ratio of the number of bits required to represent the data before compression to the number of bits required to represent the data after compression. This ratio is called the compression ratio. Suppose storing an image made up of a square array of 256 × 256 pixels requires 65,536 bytes. The image is compressed and the compressed version requires 16384 bytes. We would say that the compression ratio is 4:1.

Another way of reporting compression performance is to provide the average number of bits required to represent a single sample. This is generally referred to as the rate. For example, in the case of compressed image described above, if we assume 8 bits per byte (or pixel), the average number of bits per pixel in the compressed representation is 2. Thus we would say that the rate is 2 bits per pixel.

In lossy compression the reconstruction differs from original data. Therefore, in order to determine the efficiency of a compression algorithm, we have to have some way of quantifying the difference. The difference between the original and the reconstruction is often called the distortion.

Modeling and Coding

While reconstruction requirements may ~~differ~~ force the decision of weather a compression scheme is to be lossy or lossless, the exact compression scheme we use will depend on a number of different factors. Some of the most important factors are the characteristics of the data that need to be compressed. A compression technique that will work well for the compression of text may not work well for compressing images. Each application presents a different set of challenges.

The development of data compression algorithms for a variety of data can be divided into two phases. The first phase is usually reffered to as **modeling**. In this phase we try to extract information about any redundancy that exists in the data and describe the redundancy in the form of a model. The second phase is called **coding**. A description of a model and a "description" of how the data differ from the model are encoded, generally using a binary alphabet. The difference between the data and the model is often reffered to as the residual.

Mathematical preliminaries for Lossless Compression:

We will look at some ways to model the data that lead to efficient coding schemes.

Brief Introduction to Information Theory

The idea of a quantitative measure of information has been around for a while, the person who pulled everything together into what is now called information theory was Claude Elwood Shannon.

Shannon defined a quantity called self-information

Suppose we have an event A, which is a set of outcomes of some random experiment. If P(A) is the probability that the event A will occur, then the self information associated with A is given by

$$i(A) = \log_b \frac{1}{P(A)} = -\log_b P(A)$$

$\log(1) = 0$, and $-\log(x)$ increases as x decreases from one to zero. Therefore, if the probability of an event is ~~high~~ low, the amount of self information associated with it is high; if the probability of an event is high, the information associated with it is low.

The self information associated with the occurance of both event A and event B is

$$i(AB) = \log_b \frac{1}{P(AB)}$$

As A and B are independent

$$P(AB) = P(A)P(B)$$

$$i(AB) = \log_b \frac{1}{P(A)P(B)}$$

$$= \log_b \frac{1}{P(A)} + \log_b \frac{1}{P(B)}$$

$$= i(A) + i(B)$$

First Order Entropy

- The first order entropy is defined for a probability distribution over symbols $\{a_1, a_2, \ldots a_m\}$.

$$H = \sum_{i=1}^{m} P(a_i) \log_2 \left(\frac{1}{P(a_i)}\right)$$

- H is the average number of bits required to code up a symbol, given all we know is the probability distribution of the symbols.
- H is the Shannon lower bound on the average number of bits to code a symbol in this "source model".

Entropy examples

- $\{a, b, c\}$ with $P(A) = \frac{1}{8}$, $P(B) = \frac{1}{4}$ $P(C) = \frac{5}{8}$

$$-H = \frac{1}{8} * 3 + \frac{1}{4} * 2 + \frac{5}{8} * 0.678 = 1.3 \text{ bits/symbol}$$

- $\{a, b, c\}$ with $P(a)$ $\frac{1}{3}$, $P(b)$ $\frac{1}{3}$ $P(c) = \frac{1}{3}$. (worst case)

$$-H = 3 * \frac{1}{3} * \log_2(3) = 1.6 \text{ bits/symbol.}$$

- Note that a standard code takes 2 bits/Symbol

Symbol	a	b	c
binary code	00	01	10

Models.

Good models for sources lead to more efficient compression algorithms. In general, in order to develop techniques that manipulate data using mathematical operations. There are several approaches to building mathematical models.

Physical Models

If we know something about the physics of the data generation process, we can use that information to construct a model. For example, in speech-related applications, knowledge about the physics of speech production can be used to construct a mathematical model for the sampled speech process. Sampled speech can then be encoded using this model.

Probability Models

The simplest statistical model for the source is to assume that each letter that is generated by the source is independent of every other letter, and each occurs with the same probability. We could call this the ignorance model, as it would generally be useful only when we know nothing about the source.

The next step up in complexity is to keep the independence assumption, but remove the equal probability assumption and assign a probability of occurrance to each letter in the alphabet. For a source that generates letters from an alphabet $A = \{a_1, a_2 \cdots a_M\}$, we can have a Probability Model $P = \{P(a_1), P(a_2), \cdots P(a_M)\}$.

Markov Models

One of the most popular ways of representing dependence in the data is through the use of Markov Models, named after the Russian mathematician Andrei Andrevich Markov. for models used in lossless compression, we use a specific type of Markov process called a discrete time Markov chain. Let $\{x_n\}$ be a sequence of observations. This sequence is said to follow a kth-order Markov model if

$$P(x_n | x_{n-1}, \ldots x_{n-k}) = P(x_n | x_{n-1} \ldots x_{n-k}, \ldots)$$

In other words, knowledge of the past k symbols is equivalent to the knowledge of the entire past history of the process. The values taken on by the set $\{x_{n-1}, \ldots x_{n-k}\}$ are called the states of the process.

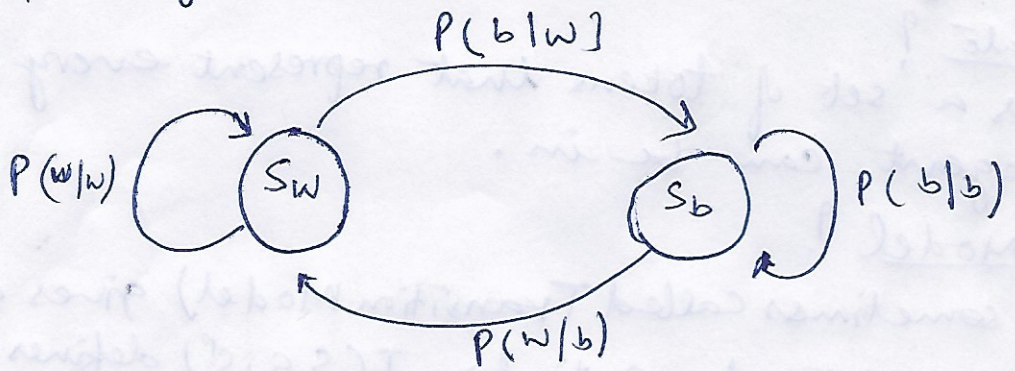


fig: A Two state Markov model for binary images

Reinforcement learning is a type of machine learning. It allows machines and software agents to automatically determine the ideal behavior within a specific context, in order to maximize its performance. Simple reward feedback is required for the agent to learn its behavior; this is known as the reinforcement signal.

There are many algorithms that tackle this issue. As a matter of fact, Reinforcement learning is defined by a specific type of problem, and all its solutions are classed as Reinforcement learning algorithms. In the problem, an agent is supposed to decide the best action to select based on his current state. When this step is repeated, the problem is known as Markov decision process.

A Markov Decision process (MDP) model contains:

- A set of possible world states S.
- A set of Models.
- A set of possible actions A.
- A real valued reward function R(s,a).
- A policy the solution of Markov Decision Process.

What is a state?

A state is a set of tokens that represent every state that the agent can be in.

What is a Model?

A model (sometimes called Transition Model) gives an action's effect in a state. In particular, T(S,a,S') defines a transition T where being in state S and taking an action 'a' takes us to state S' (S and S' may be same). For stochastic actions (noisy, non-deterministic) we also defines a probability P(S'|S,a) which represents the probability of reaching a state S' if action 'a' is taken in state S. Note Markov property states that the effects of an action taken in a state depend only that state and not on the prior history.

What is actions?

An action A is set of all possible actions. A(S) defines the set of actions that can be taken being in state S.

What is a Reward?

A Reward is a real-valued reward function. R(s) indicates the reward for simply being in the state s. R(s,a) indicates the reward for being in a state s and taking an action 'a'. R(s,a,s') indicates the reward for being in a state s, taking an action 'a' and ending up in a state s'.

What is a policy?

A policy is a solution to the Markov decision process. A policy is a mapping mapping from s to a. It indicates the action 'a' to be taken while in state s.

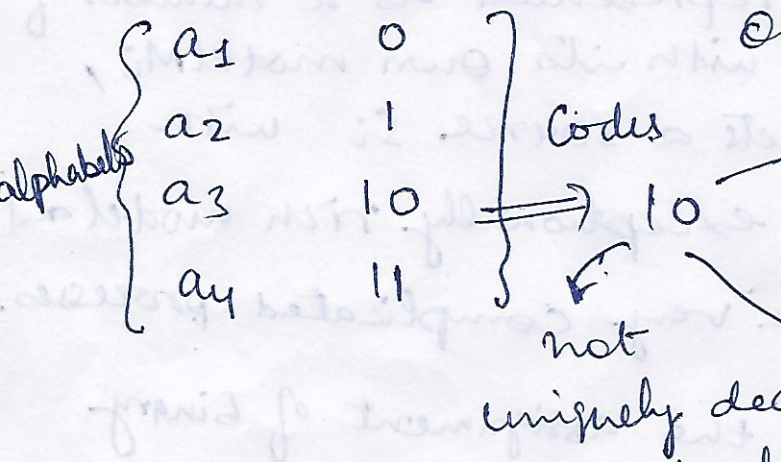
Composite source Model. Lecture 6.

In many applications, it is not easy to use a single model to describe the source. In such cases, we can define a composite source, which can be viewed as a combination or composition of several sources, with only one source being active at any given time. A composite source can be represented as a number of individual sources S_i, each with its own model M_i, and a switch that selects a source S_i with probability P_i. This is an exceptionally rich model and can be used to describe some very complicated processes.

Coding: Coding means the assignment of binary sequences to elements of an alphabet. The set of binary sequences is called a code, and the individual members of the set are called codewords. An alphabet is a collection of symbols called letters. For example the alphabet used in writing most books consists of the 26 lowercase letters, 26 uppercase letters, and a variety of punctuation marks. The ASCII code for the letter a is 1000011, the letter A is coded as 1000001, and the letter , (coma) is coded as 0011010. Notice that the ASCII codes uses the same number of bits to represent each symbol. Such a code is called a fixed length code. If we want to reduce the number of bits required to represent different messages, we need to use a different numbers of bits to represent different symbols.

If we use fewer bits to represent symbols that occur more often, on the average ~~we would~~ we would use fewer bits per symbol. The average number of bits per symbol is often called the rate of the code.

Uniquely Decodable Codes - Sequence of codes that can be decoded in only one way.



uniquely decodable because it is having more than one way to decode.

a_1	0
a_2	10
a_3	110
a_4	111

$110 \Rightarrow 10 \Rightarrow a_2 a_1 \Rightarrow$ UDC (uniquely decodable code)

1. Prefix

a - 010

b - 01011 → * Dangling suffix

* a is the prefix of b.

example 01011

(0, 01, 010, 0101, 01011)

prefix

→ Algorithm to test Uniquely Decodable Code (UDC)

1. Group the Codewords (set)
2. if no prefix found then it is UDC

   else if prefix found

   then add the dangling suffix (DS) to the set

3. if DS = same Codeword ⇒ not UDC (already present)

   else if all DS are over then it is UDC

a_1 0
a_2 1
a_3 01
a_4 11

{ 0, 1, 01, 11 }

DS = 1 (if we compare a_1 & a_3, 0 is prefix and 1 is the DS.)

{ 0, 1, 01, 11, 1 }
↳ D.S. ⇒ It is same as Codeword
So it is not UDC

a_1 0

a_2 10

a_3 110

a_4 111

{ 0, 10, 110, 111 } there is no prefix so it ⇒ UDC

2. Prefix Code

→ if any codeword is not prefix to any other codeword then it is prefix code.

* If all letters are at external node of the tree then it is prefix code. If some at external and some at internal node then it is not prefix code.

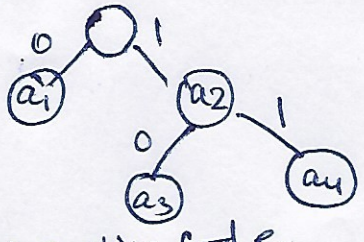
example 1.

0
10
110
111
} Prefix Code

example 2

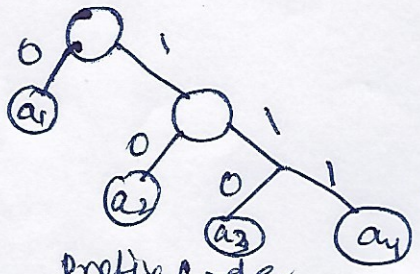
0
1
10
11
} not prefix code

1. 0 – a_1
1 – a_2
10 – a_3
11 – a_4



not prefix code

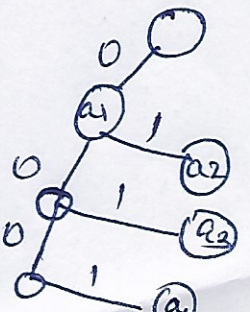
2. 0 – a_1
10 – a_2
110 – a_3
111 – a_4



prefix code

3.

0
01
001
0001



not prefix code