ETI 2507 DIGITAL IMAGE PROCESSING

Image Coding

Khamis Bamama

LECTURE OUTLINE

1. A recap on previous lecture
2. Data Compression
3. Compression methods
4. Specific family of methods – Entropy coding:

   – Transform coding/predictive coding:
5. Information theory basics

Why Compression?

- In the digital age, vast amounts of data are generated daily.
- Storage and transmission of this data pose challenges.
- Compression is crucial for efficient use of resources.

Challenges in Data Handling

- **Storage:** Large files consume significant disk space.
- **Transmission:** Sending or streaming uncompressed data takes longer.
- **Bandwidth:** Limited bandwidth affects real-time applications.
- **Cost:** Storing and transmitting large amounts of data can be expensive.

Benefits of Compression

- **Reduced Storage Space:** Compression minimizes the space needed to store data.
- **Faster Transmission:** Smaller files transmit more quickly over networks.
- **Cost Savings:** Efficient use of resources lowers storage and bandwidth costs.
- **Improved Performance:** Faster access to compressed data enhances overall system performance.

Lossless vs. Lossy Compression

- **Lossless Compression:** Preserves all image details. Examples include PNG and GIF.
- **Lossy Compression:** Sacrifices some details for higher compression. Examples include JPEG and MPEG.

Quantization

- Quantization is a fundamental step in many compression algorithms.
- It involves mapping a range of continuous values to a finite set of discrete values.
- The role of quantization is crucial in achieving compression while balancing quality.

Quantization in Image Compression

- **Color Quantization:** Reducing the number of colors in an image.
- **Spatial Quantization:** Reducing the precision of pixel values.

Quantization Example

Figure: Top left: original image; top right: 1 bit/pixel image; bottom left: 2 bits/pixel; bottom right: 3 bits/pixel.

Trade-off: Compression vs. Quality

- **Higher Quantization:** More compression, lower quality.
- **Lower Quantization:** Less compression, higher quality.
- **Balancing Act:** Choosing an appropriate level of quantization for the desired trade-off.

Compression measures

- Compression measures evaluate the performance of compression algorithms.
- These metrics help assess the trade-off between compression efficiency and the quality of the reconstructed data.
- Commonly used measures provide insights into aspects such as compression ratio, distortion, and information loss.

Compression Ratio

- **Definition:** The ratio of the size of the uncompressed data to the size of the compressed data.
- **Formula:** Compression Ratio (CR) $= \frac{\text{Original Size}}{\text{Compressed Size}}$.
- **Example:** 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 16,384 bytes. We would say that the compression ratio is 4:1.
- **Interpretation:** Higher compression ratio indicates more efficient compression.

Distortion Measures

- **Mean Squared Error (MSE):** Measures the average squared difference between original and reconstructed values.
- **Peak Signal-to-Noise Ratio (PSNR):** Describes the quality of the reconstruction by comparing it to the original signal.
- **Structural Similarity Index (SSI):** Evaluates perceived image quality, considering luminance, contrast, and structure.

Information Loss

- **Entropy:** Measures the average amount of information needed to represent a random variable.
- **Rate-Distortion Theory:** Analyzes the trade-off between the rate of information transmission and the distortion introduced during compression.

Applications

- **Image Compression:** Evaluating visual quality and file size reduction.
- **Audio Compression:** Assessing fidelity and compression efficiency.
- **Video Compression:** Balancing quality and compression ratios.

Entropy

- **Entropy:** A fundamental concept in information theory.
- **Definition:** A measure of the amount of uncertainty or disorder in a set of data.
- **Origin:** Introduced by Claude Shannon in the 1940s.
- **Interpretation:** Measures the average "surprise" or "information content" associated with an event.

Mathematical Formulation

- **Formula:** $H(X) = -\sum_i P(x_i) \log_2 P(x_i)$
- **Example:** Given an alphabet $p(a_i)$ $=\{0.25,0.25,0.2,0.15,0.15\}$

  The entropy is:

$$H = -(2\times0.25\times\log_2(0.25) - 0.2\times\log_2(0.2) - (2\times0.15\times\log_2(0.15)$$

$= 2.2855$ bits/symbol

Entropy in Compression

- **Compression Efficiency:** High entropy implies more unpredictability and, consequently, lower compression efficiency.
- **Redundancy:** Low entropy indicates redundancy, which can be exploited for compression.

Types of Entropy Coding Techniques

- **Huffman Coding:** An entropy-based coding technique that assigns shorter codes to more probable symbols.
- **Arithmetic Coding:** An entropy-based coding technique that takes a stream of input symbols and replaces it with a single floating point output number.

Huffman Coding

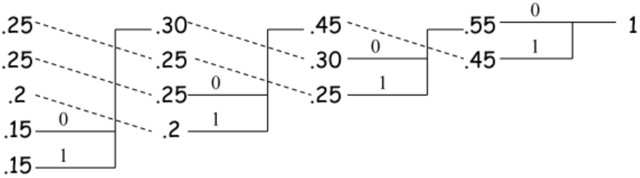
- Huffman coding is a compression algorithm.
- Developed by David A. Huffman.
- Variable-length prefix coding.
- Optimal for a given set of probabilities.
- Huffman coding is based on the frequency of symbols.
- More frequent symbols get shorter codes.

Steps of Huffman Coding

1. Let each symbol be a leaf node of a one-sided binary tree.
2. Initially, no node has a parent node.
3. Join two nodes with smallest probabilities and without parent nodes into a parent node whose probability is the sum of those of its children nodes.
4. Repeat above step until all nodes have a parent node. Starting from the root of the tree, assign one branch with 0 and other with 1.
5. Assign prefixed binary code to each symbol.

Huffman Coding Example

Symbol	a	b	c	d	e
Probability	0.25	0.25	0.2	0.15	0.15
Huffman code	01	10	11	000	001



Huffman Decoding

- Taking the example in the previous slide. If at the receiver we have the sequence {011101010011110}.
- Can be uniquely decoded as {acaaecb}.

Arithmetic Coding

- Arithmetic coding is a form of entropy encoding.
- Represents entire messages with a single real number.
- Used in data compression applications.
- Non block - doesn't generate individual codes for individual characters but treats a section of a message as a whole for encoding based on the probability of the next character.

Interval Representation

- Arithmetic coding represents a message by an interval on the real number line.
- The interval is divided and narrowed down based on the probabilities of symbols.

Steps

- Expand the range of the first letter interval of the message.
- Divide the probability distribution into the number of characters in the alphabet.
- Calculate the range by taking d = the upper bound - the lower bound
- Range of symbol = lower limit : lower limit + d(probability of symbol)
- **Example:** For a five-symbol message $a_1 a_2 a_3 a_3 a_4$ from a four-symbol source whose symbol probabilities are $P(a_1) = P(a_2) = P(a_4) = 0.2, P(a_3) = 0.4$, the arithmetic code is obtained as shown below.

Arithmetic Coding Example

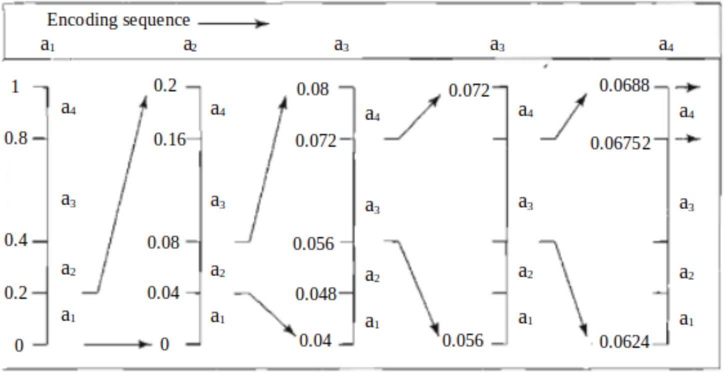


Figure: Arithmetic coding procedure

Arithmetic Coding Example

- The interval in which the tag for the sequence $a_1 a_2 a_3 a_3 a_4$ resides is $\{0.0688 \text{ , } 0.06752\}$
- As such, any member of this interval can be used as a tag.
- One popular choice is the lower limit of the interval in this case 0.0688; another possibility is the midpoint of the interval i.e 0.06816

Decoding Procedure

- Find the first symbol in the message by seeing which symbol owns the code space that the encoded message falls in.
- Since the number 0.06816 (taking the midpoint) falls between 0 and 0.2, the first character must be a_1.
- Next remove a_1 from the encoded number by subtract the low value of a_1 from the number, giving 0.06816
- Then divide 0.06816 by the range of a_1, which is 0.2. This gives a value of 0.3408
- Next, calculate where that lands in the initial subintervals of the open interval [0,1), which in this case is in the range of the next letter a_2 i.e (0.2 0.4). Repeat the above for the remaining symbols

Introduction

- **Transform Coding:** A technique used in signal and image compression.
- **Idea:** Transform the signal from its original domain to another domain where it is more efficiently represented.
- **Motivation:** Exploits the signal's energy concentration in a smaller number of transformed coefficients.

Key Concepts

- **Transform:** Mathematical operation that converts the signal from one representation to another.
- **Transformation Matrix:** Matrix used to perform the transformation.
- **Inverse Transform:** Allows reconstruction of the original signal from the transformed coefficients.

Common Transforms

- **Discrete Fourier Transform (DFT):** Used in audio and image compression.
- **Discrete Cosine Transform (DCT):** Widely used in image and video compression (e.g., JPEG).
- **Wavelet Transform:** Useful in various applications, including JPEG2000.

Discrete Cosine Transform (DCT)

- **Basis Functions:** Cosine functions of different frequencies.
- **Energy Concentration:** Most signal energy is often concentrated in a few low-frequency coefficients.
- **Applications:** JPEG compression for images, MP3 compression for audio.

Wavelet Transform

- **Localization:** Wavelet transform provides both frequency and spatial localization.
- **Multi-Resolution Analysis:** Breaks down the signal into different frequency components at various resolutions.
- **Applications:** JPEG2000 for images, video compression, and various signal processing tasks.

Advantages of Transform Coding

- **Energy Concentration:** Efficiently represents the signal in fewer coefficients.
- **Decorrelation:** Transforms often decorrelate the signal, reducing redundancy.
- **Compression Ratios:** Achieves higher compression ratios compared to direct coding.