A Very Quick Introduction to Data Compression

Jim Diamond

Outline

- Overview
- Information Theory
- Data Compression
- Example Compression Techniques
  - Huffman
  - LZW

Overview: 1

- There are two types of data compression
  - Lossless data compression
    - decompression reproduces the original data exactly
  - Lossy data compression
    - decompression reproduces an approximation of the original data
- Lossy compression (sometimes) suitable for
  - image data
  - movie data
  - audio data
- Lossy compression not so good for
  - program source files
  - novels
  - financial data

Overview: 2

- Model:
  - finite set of symbols $S = \{s_1, s_2, \ldots, s_n\}$
    - example: ASCII character set
  - "sender" wishes to transmit a string of these symbols to "receiver"
    - example: "*Today is Friday*"

$$\textit{Today} \longleftarrow \boxed{\text{Decoder}} \longleftarrow\!\!\!-\!\!\!-\!\!\!-\!\!\!-\!\!\!-\!\!\!-\!\!\!-\!\!\!-\!\!\!- \boxed{\text{Encoder}} \longleftarrow \textit{is Fr}\ldots$$

  - Communication system can only transmit two symbols "**0**" and "**1**" (*i.e.*, bits)
- Objective: minimize number of bits transmitted **while preserving** the original message.

Information Theory: 1

- Information content of a symbol is equivalent to the amount of "surprise" one experiences upon receiving it
  - example: message starts "*Frida*"
    - little surprise if next symbol is "*y*"
    - much surprise if next symbol is "*q*"
- More information is transmitted by an "unlikely" symbol than by a "likely" one
  - example above: after *Frida* has been sent, could transmit either "**0**" or "**1***q*"
- The expected symbol can be transmitted with fewer bits!

Information Theory: 2

- Example

  Walk up to someone and ask them to complete this sentence:

  “Peter Piper picked a peck of pickled ________”

  They will look you straight in the eye and say:

Information Theory: 2

- Example

  Walk up to someone and ask them to complete this sentence:

  "Peter Piper picked a peck of pickled _________"

  They will look you straight in the eye and say:

  "0"

Data Compression

- Less bits needed for expected (*i.e.*, probable) symbols
- Expectation based upon knowledge of sender and receiver
  - non-English-speaking person unable to decode **0** in examples
- Compression requires:
  - model of the data
  - non-uniform probability distribution of next symbol to be transmitted
- If, given "all" knowledge,

$$P(s_i) = P(s_j),\ \forall i, j \in \{1 \ldots n\}$$

then need, on average, at least $\log_2(n)$ bits to send next symbol

Huffman Coding: 1

- Concept: replace each (fixed-length) symbol s_i with a variable-length bit string b_i, transmit b_i instead of s_i
- Assume each symbol s_i has a certain probability p_i of being transmitted
- More probable symbols are assigned shorter bit strings
- Example:
  - $S = \{a, b, c, d\}$, $p_a = 1/2$, $p_b = 1/4$, $p_c = 1/8$, $p_d = 1/8$
  - $b_a = 0$, $b_b = 10$, $b_c = 110$, $b_d = 111$
  - straightforward coding of S requires $2m$ bits to send a message with m symbols
  - Huffman coding requires

$$1 \times \frac{m}{2} + 2 \times \frac{m}{4} + 3 \times \frac{m}{8} + 3 \times \frac{m}{8}$$

    $(= 7m/4$ bits on average)
  - 12.5% saving

Huffman Coding: 2

- Need to know probabilities:
  - a) approximate by examining a large set of messages
  - b) examine entire message before sending
    - then must also transmit frequency distribution to sender
  - c) examine (large) initial portion of message
- Other related possibilities
  - start with uniform distribution, adapt it as symbols are seen
  - cope with non-stationarity by periodically re-computing distribution
  - use 2nd order probabilities $p_{i|j}$: the probability of seeing s_i given that the previous symbol was s_j

Lempel–Ziv (& sometimes Welch): 1

- Concept:
  - create a fixed-size "dictionary" of common sub-strings of the message
  - replace these **variable-length** sub-strings of symbols with **fixed-length** bit-strings
  - these bit-strings are pointers into the dictionary
- Compression achieved if there enough frequently occurring patterns in the message
- This technique works well on messages such as English text

Lempel–Ziv (& sometimes Welch): 2

- Horribly Simplified Example: suppose the dictionary contains

```
...
774: "cat"
775: "catatonic"
776: "catastrophic"
777: "dog"
778: "dogma"
779: "dogmatic"
...
```

and the next piece of input to be compressed is

```
catastrophic rational dogmatic ...
```

then the very next compressor output would be

```
776
```

Now on to XML