Audio and Video Coding

Keith Winstein (keithw@mit.edu)

Student Information Processing Board

November 6, 2008

Overview

1. Mathematical preliminaries
2. Lossless compression
3. Audio coding
4. Video coding

The problems:

- **Representing information digitally.** We have Dvorak's "New World Symphony." How should we represent this inside a computer? $\Rightarrow$ *Clearly a domain-specific problem...*
- **Lossless compression.** We have a set of bitstrings, like the great works of English literature represented in UTF-16LE. What's the most efficient way to represent them?
- **Lossy compression.** We have a set of bitstrings, and we want to represent them efficiently, but we're willing to tolerate some distortion. Given a "distortion function" that relates two bitstrings, and a maximum tolerance for distortion, what's the most efficient way to represent the strings? Or the constraint can be on maximum bits, trying to minimuze distortion.

The problems:

- **Representing information digitally.** We have Dvorak's "New World Symphony." How should we represent this inside a computer? $\Rightarrow$ *Clearly a domain-specific problem...*
- **Lossless compression.** We have a set of bitstrings, like the great works of English literature represented in UTF-16LE. What's the most efficient way to represent them?
- **Lossy compression.** We have a set of bitstrings, and we want to represent them efficiently, but we're willing to tolerate some distortion. Given a "distortion function" that relates two bitstrings, and a maximum tolerance for distortion, what's the most efficient way to represent the strings? Or the constraint can be on maximum bits, trying to minimuze distortion.

These are inevitably domain-specific too!

Say we want to transmit a sequence of numbers.

9	11	11	11	14	13	15	17	16	17	20	21	...

Say we want to transmit a sequence of numbers.

9	11	11	11	14	13	15	17	16	17	20	21	...

One possibility: send five bits per symbol.

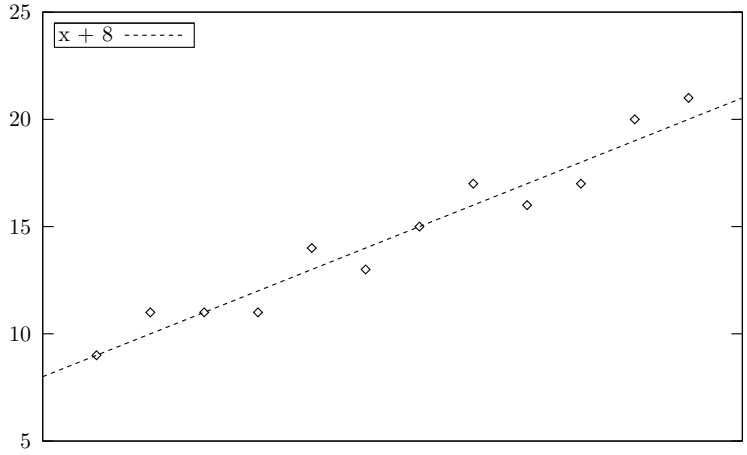
Or:

Say we want to transmit a sequence of numbers.

9	11	11	11	14	13	15	17	16	17	20	21	. . .

One possibility: send five bits per symbol.

Or:

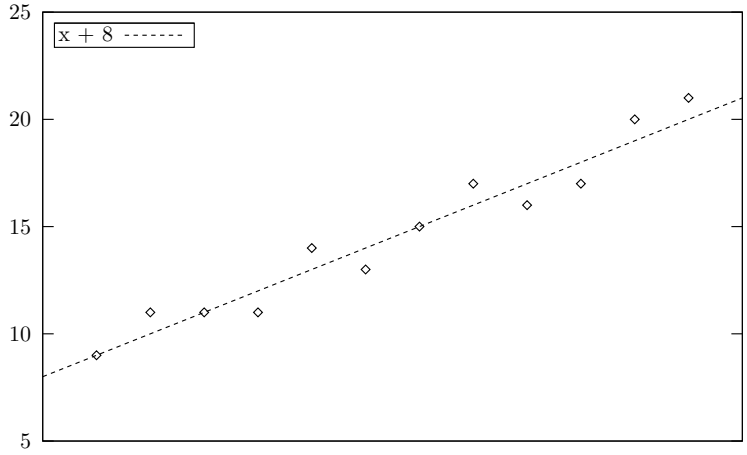


Say we want to transmit a sequence of numbers.

9	11	11	11	14	13	15	17	16	17	20	21	. . .

One possibility: send five bits per symbol.

Or:



0	1	0	-1	1	-1	0	1	-1	-1	1	1	. . .

Subtract $x + 8$, then send just two bits per symbol!

Sayood, 1996

Another example:

27	28	29	28	26	27	29	28	30	32	34	36	38	...

Send six bits per symbol?

Another example:

27	28	29	28	26	27	29	28	30	32	34	36	38	...

Send six bits per symbol?

This time we can subtract $f[x-1]$:

♡	1	1	-1	-2	1	2	-1	2	2	2	2	2	...

Sayood, 1996

So **modeling** the source of data to let us **predict** symbols is important to finding an efficient representation.

How low can we go?

Shannon (1948) showed that the best that lossless compression can do, on average, is to match the *entropy* of the data source.

Let X be a discrete random variable, which outputs symbol $x \in \mathcal{X}$ with probability $p(x)$. Then the *entropy* $H(X)$ is:

$$H(X) = \sum_{x \in \mathcal{X}} p(x) \log \frac{1}{p(x)}.$$

Self-information

Entropy:

$$H(X) = \sum_{x \in \mathcal{X}} p(x) \log \frac{1}{p(x)} = -\sum_{x \in \mathcal{X}} p(x) \log p(x) \qquad (1)$$

Each possible output $x \in \mathcal{X}$ will occur with a certain probability. The more probable outputs are not that surprising. The less probable outputs have a high "surprise factor," also known as *self-information*.

We define the self-information of symbol x as $\log \frac{1}{p(x)}$.

Entropy is just the expected "surprise" per symbol! In other words, the entropy of the source is the average self-information per symbol.

Benefits of a logarithmic measure.

- Why do we define entropy *logarithmically*, as $-\sum p(x) \log p(x)$?

  Because it works. We want the information in two independent sources to be the sum of the information in each independent source. With a logarithmic measure, $H(X, Y) = H(X) + H(Y)$ for independent sources.

- What do we call the measure of information?

  With a base-2 logarithym, *bits*. A natural logarithmy gives *nats*. Base 10 yields *hartleys* (nobody uses hartleys).

Huffman coding.

- In coding data from a source, we want the coded rate to match the entropy as closely as possible. Each symbol's code length should be close to its self-information. More "surprising" symbols should require more bits.

Huffman coding.

- In coding data from a source, we want the coded rate to match the entropy as closely as possible. Each symbol's code length should be close to its self-information. More "surprising" symbols should require more bits.
- David Huffman came up with a way of generating an optimal code as a homework project at MIT. (It was the first class in information theory ever, taught by Professor Fano.)

Huffman coding.

- In coding data from a source, we want the coded rate to match the entropy as closely as possible. Each symbol's code length should be close to its self-information. More "surprising" symbols should require more bits.
- David Huffman came up with a way of generating an optimal code as a homework project at MIT. (It was the first class in information theory ever, taught by Professor Fano.)
- The code assigns rarer symbols more bits, and the most common symbols are coded with the fewest bits. The procedure is based on one rule, applied recursively:
  1. The two symbols that occur least frequently should have the same length and differ only in the last bit.

Huffman coding.

- In coding data from a source, we want the coded rate to match the entropy as closely as possible. Each symbol's code length should be close to its self-information. More "surprising" symbols should require more bits.
- David Huffman came up with a way of generating an optimal code as a homework project at MIT. (It was the first class in information theory ever, taught by Professor Fano.)
- The code assigns rarer symbols more bits, and the most common symbols are coded with the fewest bits. The procedure is based on one rule, applied recursively:
  1. The two symbols that occur least frequently should have the same length and differ only in the last bit.
- As an added benefit, Huffman codes are *prefix codes*, meaning we can decode each symbol immediately, without having to "look ahead."

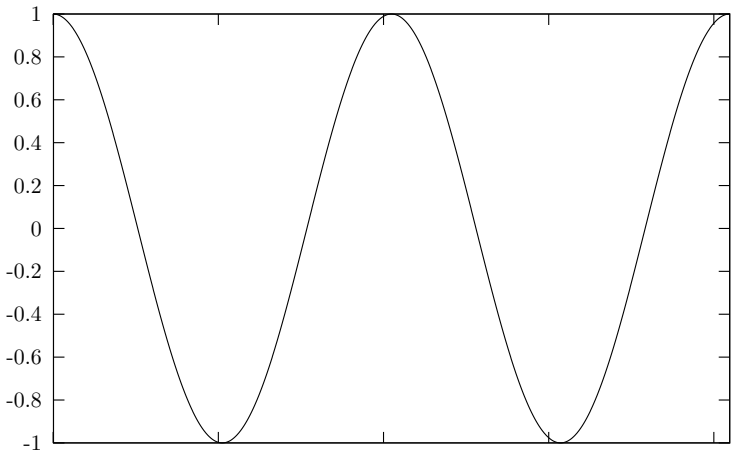
Huffman coding example.

' ' $\Rightarrow$ 111	E $\Rightarrow$ 000
T $\Rightarrow$ 1101	A $\Rightarrow$ 1011
I $\Rightarrow$ 1001	O $\Rightarrow$ 1000
R $\Rightarrow$ 0111	S $\Rightarrow$ 0110
N $\Rightarrow$ 0100	H $\Rightarrow$ 11001
C $\Rightarrow$ 10101	L $\Rightarrow$ 10100
D $\Rightarrow$ 01011	M $\Rightarrow$ 00111
U $\Rightarrow$ 00110	P $\Rightarrow$ 00100
F $\Rightarrow$ 110001	G $\Rightarrow$ 110000
B $\Rightarrow$ 010100	W $\Rightarrow$ 001011
Y $\Rightarrow$ 001010	V $\Rightarrow$ 0101010
K $\Rightarrow$ 01010110	X $\Rightarrow$ 010101110
Q $\Rightarrow$ 0101011110	J $\Rightarrow$ 01010111110
Z $\Rightarrow$ 01010111111	

Example by hand.

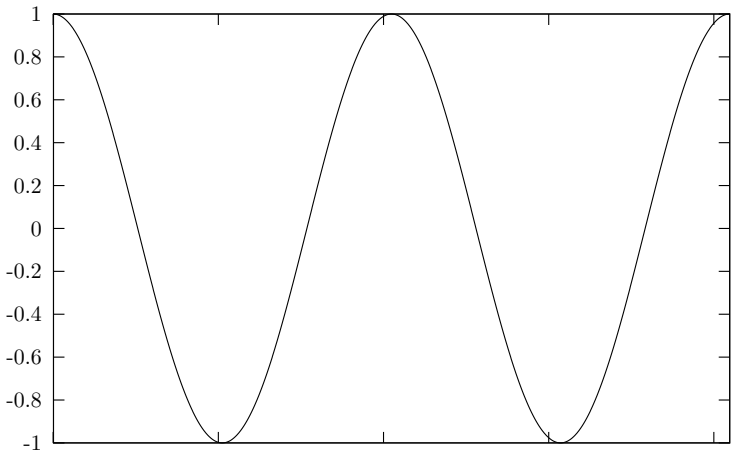
Coding audio.

What we want:

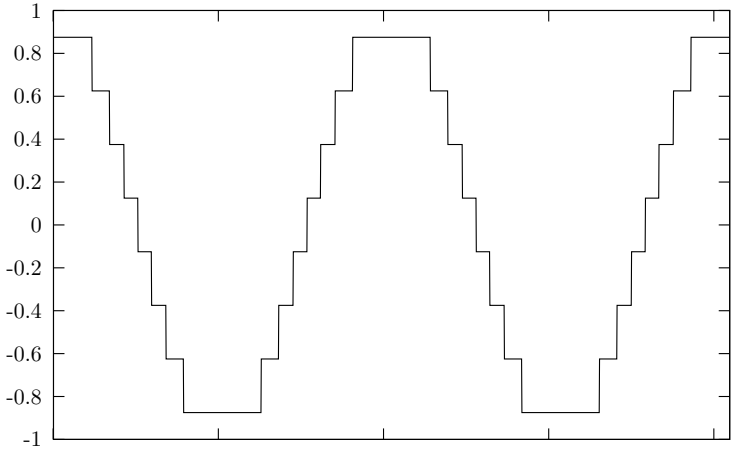


Coding audio.

What we want:



One way to compress:



Coding audio.

A better idea: put our bits where they can do the most good.

- Nonlinear quantizer – pack output values closer together for more probably outputs.

Coding audio.

A better idea: put our bits where they can do the most good.

- Nonlinear quantizer – pack output values closer together for more probably outputs.
- Code multiple symbols at once to increase efficiency (vector quantization).

Coding audio.

A better idea: put our bits where they can do the most good.

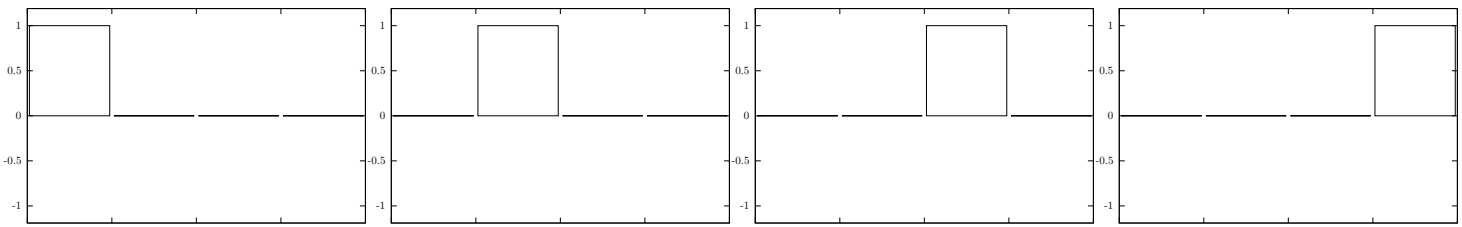
- Nonlinear quantizer – pack output values closer together for more probably outputs.
- Code multiple symbols at once to increase efficiency (vector quantization).
- **Use a smart distortion measure that makes use of the ear's weakenesses.**

Change of basis.

- Just like we were able to increase efficiency on the number sequences by changing the basis, we can do that to help with audio signals.

Change of basis.

- Just like we were able to increase efficiency on the number sequences by changing the basis, we can do that to help with audio signals.



Change of basis.

- Just like we were able to increase efficiency on the number sequences by changing the basis, we can do that to help with audio signals.