Lecture 21

Friday, April 03, 2015 10:47 AM

Huffman : Lossless Compression

We saw : $H(X) \leq E L_H(X) < H(X) + 1$ bits

One bit gap (overhead)

Encode over blocks:
1. The overhead becomes negligible.
2. Take advantage of correlation.

(the)(e)(ang)(ry)(bi)(rd) e.g. Code : aa → 00100, ab → 010, ⋮

Example without correlation:

$\Omega = \{H, T\}$, $Prob(X=H) = 0.89$, $Prob(X=T) = 0.11$

$H(X) = \frac{1}{2}$ bit

Huffman code: Code: H → 0, T → 1 $E L_u(X) = 1$

Encode over pairs: $(X_1, X_2) \in \Omega$, $\Omega = \{HH, HT, TH, TT\}$

$p(HH) = (0.89)^2 = 0.7921$

$p(HT) = 0.89 \cdot 0.11 = 0.0979$ by independence

$p(TH) =$ ↓

$p(TT) = (0.11)^2 = 0.0121$

$C(X_1, X_2)$						
0	HH	0.7921	0.7921	0.7921	0 / 1	1
11	HT	0.0979	0.11	0.2079	0 / 1	
			0.0979			

0	HH	
11	HT	0.0979
100	TH	0.0979
101	TT	0.0121

0.11 (0, 0.0979)

0.2079 (0, 1)

$$E L_H(X) = 0.7921 \cdot 1 + 0.0979 \cdot 2 + 0.11 \cdot 3 \approx 1.3 \text{ bits}$$

$$\text{bits/symbol} \approx \frac{1.3}{2} = 0.65 \text{ bits}$$

Bounds for longer blocks:

Let L_n^* be the normalized expected length of the Huffman code for blocklength n.

$$L_n^* = \frac{1}{n} E L_H(X_1 X_2 \ldots X_n)$$

$$n H(X) = H(X_1 X_2 \ldots X_n) \leq n L_n^* < H(X_1 X_2 \ldots X_n) + 1 \text{ bit} = n H(X) + 1 \text{ bit}$$

↑ Assuming independent and identically distributed

$$\Rightarrow \quad H(X) \leq L_n^* < H(X) + \frac{1}{n} \text{ bits.}$$

In general, "entropy rate" is the fundamental compression limit.

JPEG:

Color is 3-dimensional as perceived by the human eye.

red, green, blue ← basis
red, yellow, blue ←

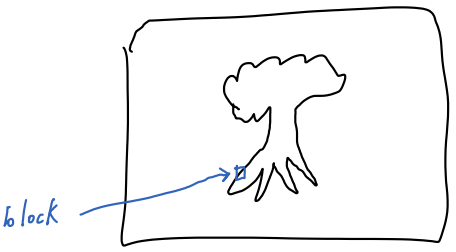
Y, C_B, C_R Y is intensity

Y, C_B, C_R Y is intensity

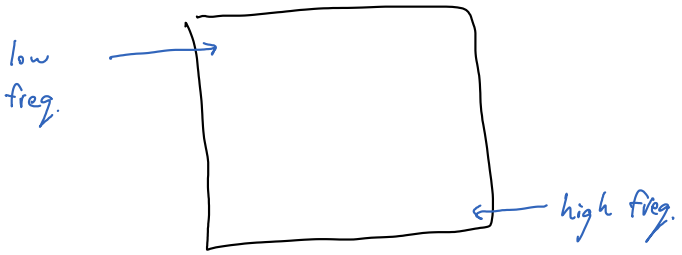
C_B, color } Downsample by 2

C_R, color }

Split in 8×8 pixel blocks:



DCT: (think of as DFT)



Uniform Quantization:

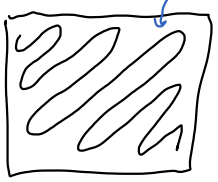
- Scale each entry (by different values)
- Round to nearest interger

16	18	
18		50
	50	100

← Matrix of scaling constants
Fixed part of the protocol.

Last steps: Lossless

Quantized DCT



Expect many zeros in a row

- Run-length code:

Example: 11111 00000001111110000

$(5,1), (7,0), (6,1), (4,0)$

$5, 7, 6, 4$

- Huffman Code

Error Detection and Correction: Add redundancy in a smart way.

- Reed-Solomon:

1960

Many applications – in much of today's technology

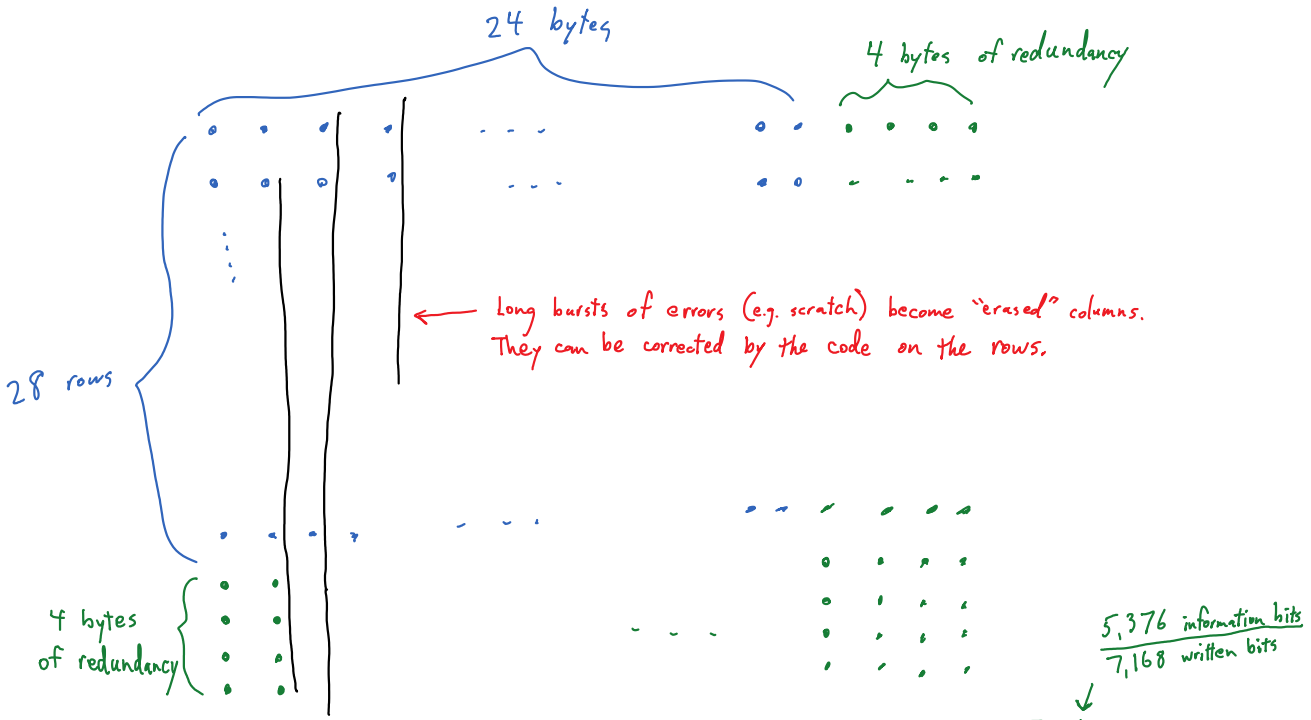
Specifics:
- Take n symbols of information
- Add t symbols of redundancy ← (used finite-field arithmatic)
- Correct errors and erasures.

$$2 \cdot N_{errors} + N_{erasures} \leq t$$

- If it cannot correct, it usually declares failure.
- Decoder can be understood with the Fourier Transform

As Seen on CD!

- symbols are bytes (8 bits)
- 2 layers with interleaving.



$$\text{Rate} : \frac{24}{28} \cdot \frac{28}{32} = \frac{3}{4}$$

$\frac{5,376 \text{ information bits}}{7,168 \text{ written bits}}$

Probability of error:
- noise model is important for analysis
- good for bursty errors
- not so good for scattered noise (15 well placed bit-errors breaks the error correction)