# 3F1 Information Theory, Lecture 3

Jossy Sayir

UNIVERSITY OF CAMBRIDGE
Department of Engineering

Michaelmas 2011, 28 November 2011

# Summary of last lecture

- Prefix-free code, rooted tree with probabilities
- Kraft's inequality:

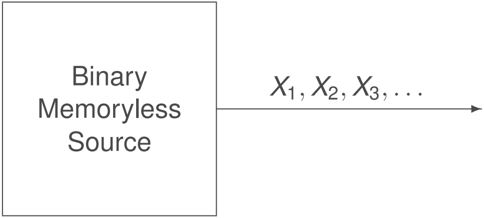
$$\sum_i D^{-w_i} \leq 1$$

- Shannon-Fano coding: $w_i = \lceil -\log_D p_i \rceil$
- Huffman's optimal code

**Performance of Shannon-Fano and Huffman coding and Shannon's converse theorem**

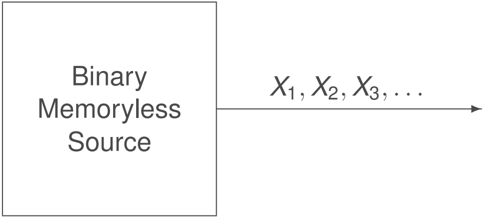
$$\frac{H(X)}{\log D} \leq E[W] < \frac{H(X)}{\log D} + 1$$

# Encoding the output of a source




- $X_1, X_2, \ldots$ independent and identically distributed
- $P_{X_i}(1) = 1 - P_{X_i}(0) = 0.01$
- What's the optimal binary variable length code for $X_i$?
- $E[W] = 1$, but $H(X) = 0.081$ bits.
- Redundancy $\rho_1 = E[W] - H(X) = 0.9190$
- Can we do better?

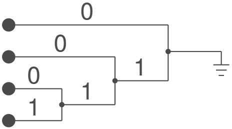
# Encoding the output of a source




- $X_1, X_2, \ldots$ independent and identically distributed
- $P_{X_i}(1) = 1 - P_{X_i}(0) = 0.01$
- What's the optimal binary variable length code for $X_i$?
- $E[W] = 1$, but $H(X) = 0.081$ bits.
- Redundancy $\rho_1 = E[W] - H(X) = 0.9190$
- Can we do better?

# Parsing a source into blocks of 2

| $x_1 x_2$ | $P_{X_1 X_2}(x_1, x_2)$ |
|---|---|
| 00 | 0.9801 |
| 01 | 0.0099 |
| 10 | 0.0099 |
| 11 | 0.0001 |




| Codeword |
|---|
| 0 |
| 10 |
| 110 |
| 111 |

- $E[W] = 1.0299$, $H(X_1 X_2) = 0.1616$ bits, where

$$H(X_1 X_2) = -\sum_{x_1 x_2} P_{X_1 X_2}(x_1, x_2) \log P_{X_1 X_2}(x_1, x_2)$$

- Redundancy per symbol:

$$\rho_2 = \frac{E[W] - H(X_1 X_2)}{2} = 0.4342$$

- Can we do better?

# Encoding a block of length $N$

- If encoding a block of length $N$ using Huffman or Shannon-Fano coding,

$$H(X_1 \dots X_N) \leq E[W] < H(X_1 \dots X_N) + 1$$

where

$$H(X_1 \dots X_2) = -\sum_{x_1 \dots x_2} P(x_1, \dots, x_2) \log P(x_1, \dots, x_2)$$

- Thus,

$$\rho_N = \frac{E[W] - H(X_1 \dots X_N)}{N} \leq \frac{H(X_1 \dots X_N) + 1 - H(X_1 \dots X_N)}{N} = \frac{1}{N}$$

and

$$\lim_{N \to \infty} \rho_N = 0,$$

i.e., the redundancy tends to zero

# The problem with block compression

- The alphabet size grows exponentially in the block length $N$
- Huffman's and Fano's algorithms become infeasible for large $N$, as does storing the codebook
- In Shannon's version of Shannon-Fano coding, the cumulative probability can be computed recursively
- Compute the cumulative probability for a specific block without computing all others!
- If only there wasn't the need for the alphabet to be ordered. . .

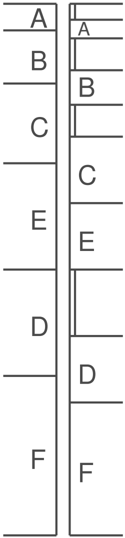
# Reminder: Shannon's version of S-F Coding

| $x$ | $P_X(x)$ | $P(X<x)$ | $P(X<x)\|_b$ | $\lceil -\log P_X(x) \rceil$ | Codeword |
|---|---|---|---|---|---|
| F | .30 | 0.0 | 0.00000000 | 2 | 00 |
| D | .20 | 0.3 | 0.01001101 | 3 | 010 |
| E | .20 | 0.5 | 0.10000000 | 3 | 100 |
| C | .15 | 0.7 | 0.10110011 | 3 | 101 |
| B | .10 | 0.85 | 0.11011010 | 4 | 1101 |
| A | .05 | 0.95 | 0.11110011 | 5 | 11110 |

- Order the symbols in order of non-increasing probability
- Compute the cumulative probabilities
- Express the cumulative probabilities in $D$-ary
- The codeword is the fractional part of the cumulative probabilities truncated to length $\lceil -\log P_X(x) \rceil$

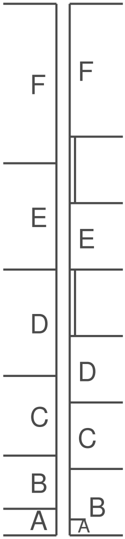
# Source intervals / code intervals

| $x$ | $P(X<x)$ | $P(X<x)\|_b$ |
|---|---|---|
| F | 0.0 | 0.00 |
| D | 0.3 | 0.010 |
| E | 0.5 | 0.100 |
| C | 0.7 | 0.101 |
| B | 0.85 | 0.1101 |
| A | 0.95 | 0.11110 |



# What if we don't order the source probabilities?

| $x$ | $P(X<x)$ | $P(X<x)\|_b$ |
|---|---|---|
| A | 0.0 | 0.00000 |
| B | 0.05 | 0.000 |
| C | 0.15 | 0.001 |
| D | 0.3 | 0.010 |
| E | 0.5 | 0.100 |
| F | 0.7 | 0.11 |



# New approach

| $x$ | $[a, b]$ |
|---|---|
| A | $[0.0, 0.05]$ |
| B | $[0.05, 0.15]$ |
| C | $[0.15, 0.3]$ |
| D | $[0.3, 0.5]$ |
| E | $[0.5, 0.7]$ |
| F | $[0.7, 1.0]$ |

$$a = P(X < x)$$
$$b = P(X < x) + P(X = x)$$


- Draw source intervals in no particular order
- Pick the largest interval $[k2^{-w_i}, (k+1)2^{-w_i}]$ that fits in each source interval
- How large is $w_i$?
- $w_i \leq \lceil -\log p_i \rceil + 1$

# New approach

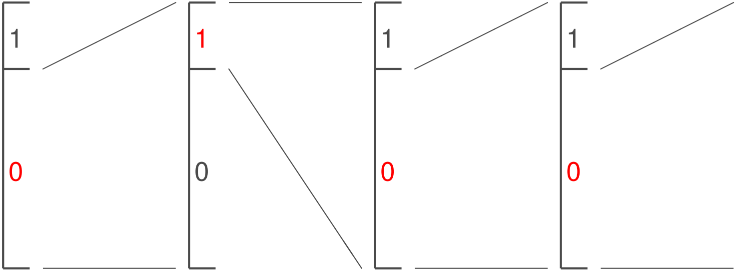
| $x$ | $[a, b]$ |
|---|---|
| A | $[0.0, 0.05]$ |
| B | $[0.05, 0.15]$ |
| C | $[0.15, 0.3]$ |
| D | $[0.3, 0.5]$ |
| E | $[0.5, 0.7]$ |
| F | $[0.7, 1.0]$ |

$$a = P(X < x)$$
$$b = P(X < x) + P(X = x)$$


- Draw source intervals in no particular order
- Pick the largest interval $[k2^{-w_i}, (k+1)2^{-w_i}]$ that fits in each source interval
- How large is $w_i$?
- $w_i \leq \lceil -\log p_i \rceil + 1$

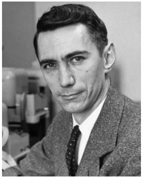
# Arithmetic Coding

- Recursive computation of the source interval:



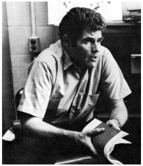

- At the end of the source block, generate the shortest possible codeword whose interval fits in the computed source interval
- $E[W] < H(X_1 \ldots X_N) + 2$

# Arithmetic Coding = Recursive Shannon-Fano Coding



Claude E. Shannon

Robert L. Fano



Peter Elias

Richard C. Pasco

Jorma Rissanen

- $H(X_1 \dots X_N) \leq E[W] < H(X_1 \dots X_N) + 2$
- $\rho_N = 2/N$
- No need to wait until all the source block has been received to start generating code symbols
- Arithmetic Coding is an infinite state machine whose states need ever growing precision
- Finite precision implementation requires a few tricks (and loses some performance)

# English Text

| Letter | Frequency | Letter | Frequency | Letter | Frequency |
|---|---|---|---|---|---|
| a | 0.08167 | j | 0.00153 | s | 0.06327 |
| b | 0.01492 | k | 0.00772 | t | 0.09056 |
| c | 0.02782 | l | 0.04025 | u | 0.02758 |
| d | 0.04253 | m | 0.02406 | v | 0.00978 |
| e | 0.12702 | n | 0.06749 | w | 0.02360 |
| f | 0.02228 | o | 0.07507 | x | 0.00150 |
| g | 0.02015 | p | 0.01929 | y | 0.01974 |
| h | 0.06094 | q | 0.00095 | z | 0.00074 |
| i | 0.06966 | r | 0.05987 | | |

- $H(X) = 4.17576$ bits
- $P(''and'') = 0.08167 \times 0.06749 \times 0.04253 = 0.0002344216$
- $P(''eee'') = 0.12702^3 = 0.00204935$
- $P(''eee'') >> P(''and'')$. Is this right? No!
- Can we do better than $H(X)$ for sources with memory?

# English Text

| Letter | Frequency | Letter | Frequency | Letter | Frequency |
|---|---|---|---|---|---|
| a | 0.08167 | j | 0.00153 | s | 0.06327 |
| b | 0.01492 | k | 0.00772 | t | 0.09056 |
| c | 0.02782 | l | 0.04025 | u | 0.02758 |
| d | 0.04253 | m | 0.02406 | v | 0.00978 |
| e | 0.12702 | n | 0.06749 | w | 0.02360 |
| f | 0.02228 | o | 0.07507 | x | 0.00150 |
| g | 0.02015 | p | 0.01929 | y | 0.01974 |
| h | 0.06094 | q | 0.00095 | z | 0.00074 |
| i | 0.06966 | r | 0.05987 | | |

- $H(X) = 4.17576$ bits
- $P(''and'') = 0.08167 \times 0.06749 \times 0.04253 = 0.0002344216$
- $P(''eee'') = 0.12702^3 = 0.00204935$
- $P(''eee'') >> P(''and'')$. Is this right? No!
- Can we do better than $H(X)$ for sources with memory?

# Entropy revisited

- We already know the symbol and block entropies:

$$H(X) = -\sum_{x} P_X(x) \log P_X(x)$$

$$H(XY) = -\sum_{xy} P_{XY}(x, y) \log P_{XY}(x, y)$$

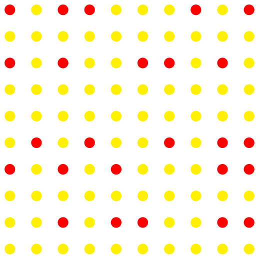
- Entropy conditioned on an event:

$$H(X|Y = y) = -\sum_{x} P_{X|Y}(x|y) \log P_{X|Y}(x|y)$$

## Equivocation or conditional entropy

$$H(X|Y) = H(XY) - H(Y) = E[H(X|Y = Y)] = \sum_{y} P_Y(y) H(X|Y = y)$$

# Does conditioning reduce uncertainty?



- $X$ color of randomly picked ball
- $Y$ row of picked ball (1-10)
- $H(X) = h(1/4) = 2 - \frac{3}{4}\log_2 3 =$ 0.811 bits
- $H(X|Y=1) = 1$ bit $> H(X)$
- $H(X|Y=2) = 0$ bits $< H(X)$
- $H(X|Y) = \frac{5\times 1 + 5\times 0}{10} = \frac{1}{2} < H(X)$

## Conditioning Theorem

$$0 \le H(X|Y) \le H(X)$$

# Discrete Stationary Source (DSS) properties

- $H_N(X) \stackrel{\text{def}}{=} \frac{1}{N} H(X_1 \ldots X_N)$

## Entropy Rate

- $H(X_N | X_1 \ldots X_{N-1}) \leq H_N(X)$
- $H(X_N | X_1 \ldots X_{N-1})$ and $H_N(X)$ are non-increasing functions of $N$
- $\lim_{N \to \infty} H(X_N | X_1 \ldots X_{N-1}) = \lim_{N \to \infty} H_N(X) \stackrel{\text{def}}{=} H_\infty(X)$
- $H_\infty(X)$ is the entropy rate of a DSS

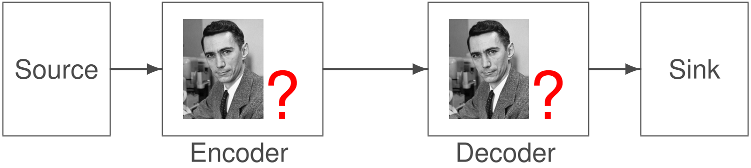
## Shannon's converse source coding theorem for a DSS

$$\frac{E[W]}{N} \geq \frac{H_\infty(X)}{\log D}$$

# Coding for discrete stationary sources

- Arithmetic coding can use conditional probabilities
- The intervals will be different at every step depending on the source context
- "Prediction by Partial Matching" (PPM) and "Context Tree Weighing" (CTW) are techniques to build the context tree based source model on the fly, achieving compression rates of approx 2.2 binary symbols per ASCII character
- What is $H_\infty(X)$ for English text? (assuming language is a stationary source, which is a disputed proposition)

# Shannon's Twin Experiment



# Shannon's twin experiment

- Shannon 1 and Shannon 2 are hypothetical fully identical twins (who look alike, talk alike, and think exactly alike)
- An operator in the transmitter asks Shannon 1 to guess the next source symbol of the source based on the context
- The operator counts the number of guesses until Shannon 1 gets it right, and transmits this number
- An operator in the receiver asks Shannon 2 to guess the next source symbol based on the context. Shannon 2 will get the same answer as Shannon 1 after the same number of guesses.
- An upper bound on the entropy rate of English is the entropy of the number of guesses
- A better bound would take dependencies between numbers of guesses into account (if Shannon 1 needed many guesses for a symbol then chances are that he will need many for the next as well, whereas if he guessed right the first time, chances are that he's in the middle of a word and will guess the next symbol correctly as well)