

T-61.182 Information Theory and Machine Learning  
**Data Compression (Chapters 4-6)**

presented by Tapani Raiko

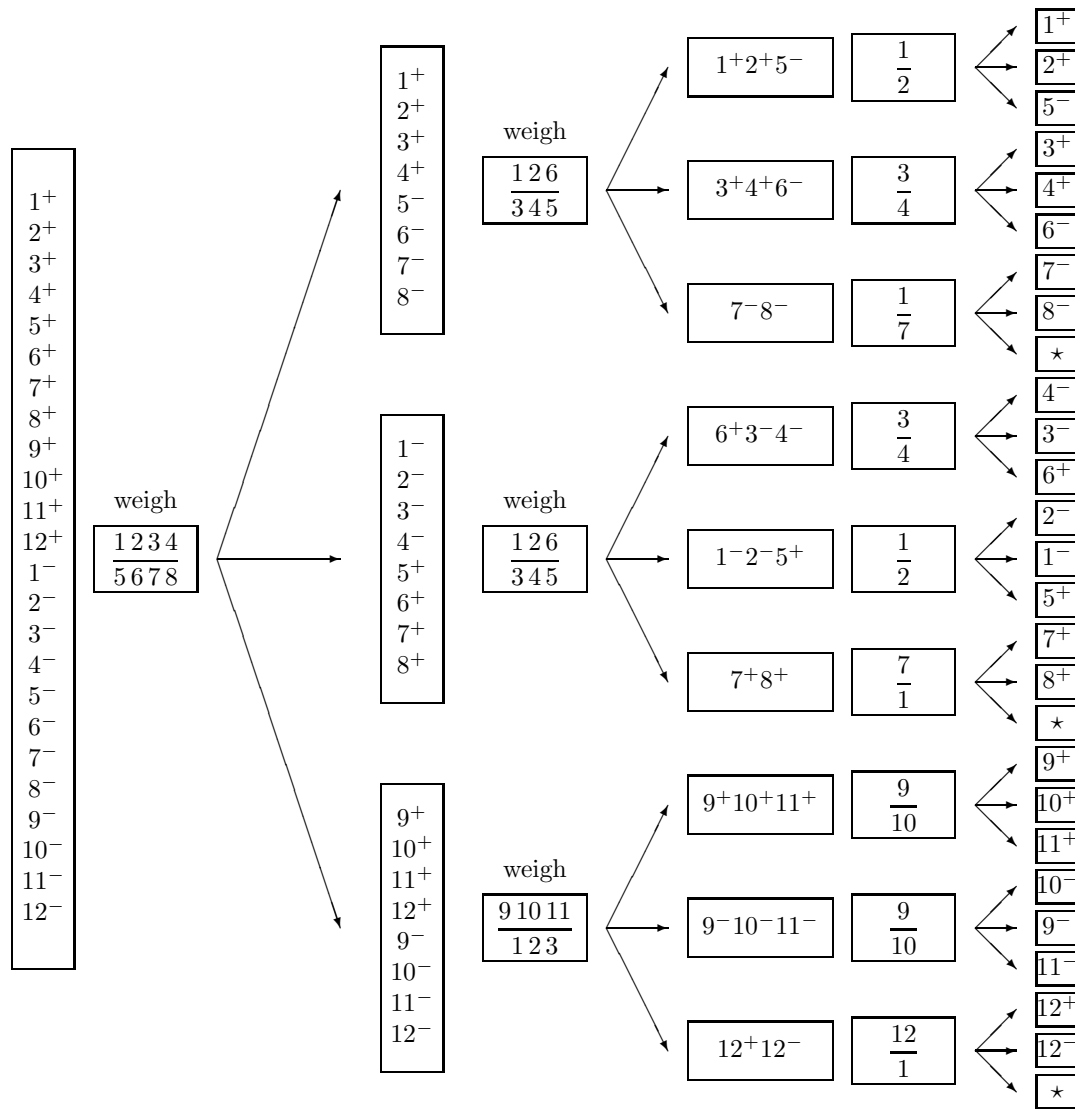
Feb 26, 2004

## Contents (Data Compression)

	Chap. 4	Chap. 5	Chap. 6
Data	Block	Symbol	Stream
Lossy?	Lossy	Lossless	Lossless
Result	Shannon's source coding theorem	Huffman coding algorithm	Arithmetic coding algorithm

## Weighting Problem (What is information?)

- 12 balls, all equal in weight except for one
- Two-pan balance to use
- Determine which is the odd ball and whether it is heavier or lighter
- As few uses of the balance as possible!
  
- The outcome of a random experiment is guaranteed to be most informative if the probability distribution over outcomes is uniform



# Definitions

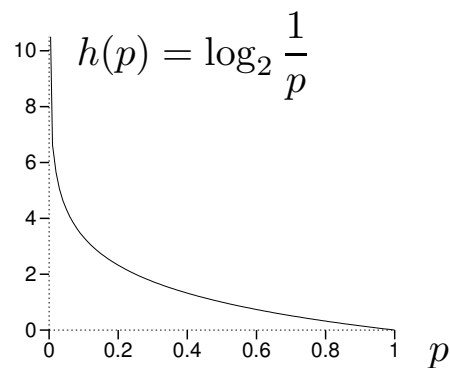
- Shannon information content:

$$h(x = a_i) \equiv \log_2 \frac{1}{p_i}$$

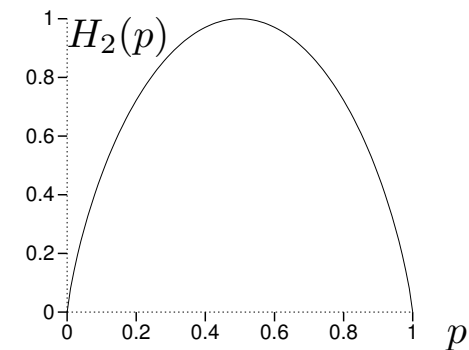
- Entropy:

$$H(X) = \sum_i p_i \log_2 \frac{1}{p_i}$$

- Both are additive for independent variables

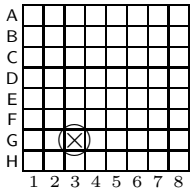
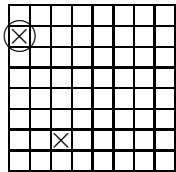
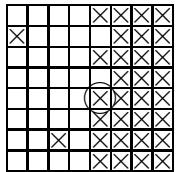
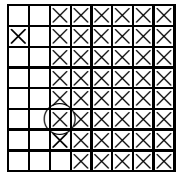
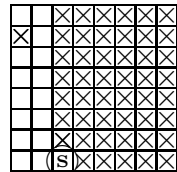


$p$	$h(p)$	$H_2(p)$
0.001	10.0	0.011
0.01	6.6	0.081
0.1	3.3	0.47
0.2	2.3	0.72
0.5	1.0	1.0



# Game of Submarine

- Player hides a submarine in one square of an 8 by 8 grid
- Another player tries to hit it

					
move #	1	2	32	48	49
question	G3	B1	E5	F3	H3
outcome	$x = n$	$x = n$	$x = n$	$x = n$	$x = y$
$P(x)$	$\frac{63}{64}$	$\frac{62}{63}$	$\frac{32}{33}$	$\frac{16}{17}$	$\frac{1}{16}$
$h(x)$	0.0227	0.0230	0.0443	0.0874	4.0
Total info.	0.0227	0.0458	1.0	2.0	6.0

- Compare to asking 6 yes/no questions about the location

## Raw Bit Content

- A binary name is given to each outcome of a random variable  $X$
- The length of the names would be  $\log_2 |\mathcal{A}_X|$   
(assuming  $|\mathcal{A}_X|$  happens to be a power of 2)
- Define: The raw bit content of  $X$  is

$$H_0(X) = \log_2 |\mathcal{A}_X|$$

- Simply counts the possible outcomes - no compression yet
- Additive:  $H_0(X, Y) = H_0(X) + H_0(Y)$

# Lossy Compression

- Let

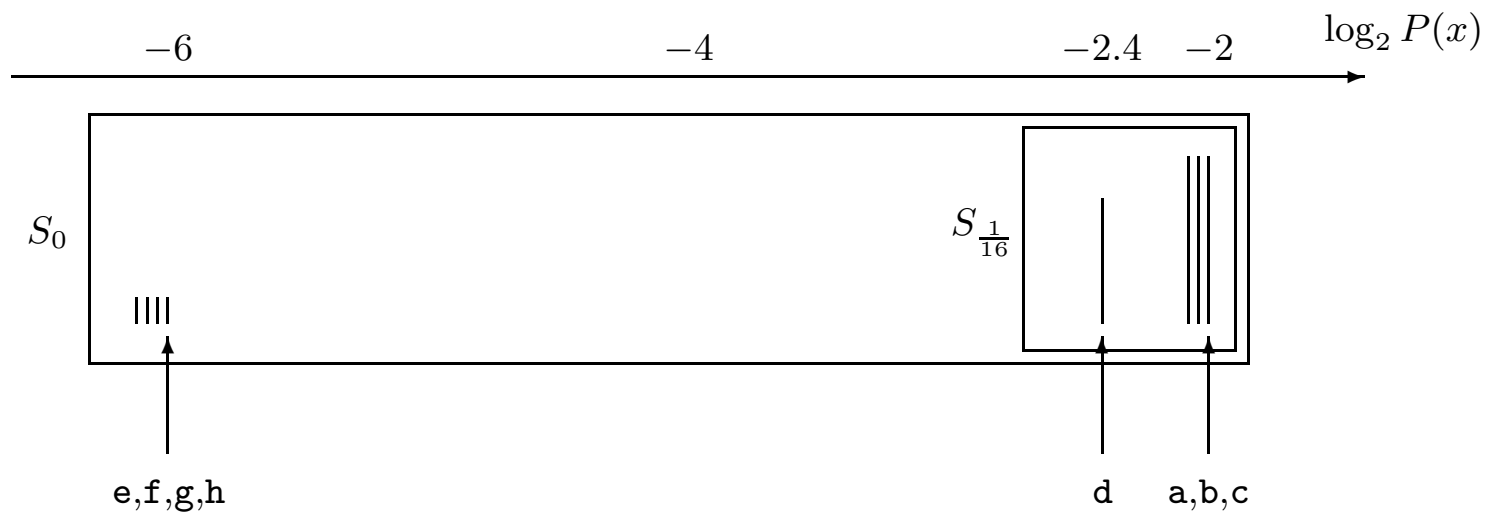
$$\mathcal{A}_X = \{ a, b, c, d, e, f, g, h \}$$

$$\mathcal{P}_X = \left\{ \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{3}{16}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64} \right\}$$

- The raw bit content is 3 bits (8 binary names)
- If we are willing to run a risk of  $\delta = 1/16$  of not having a name for  $x$ , then we can get by with 2 bits (4 names)

$\delta = 0$		$\delta = 1/16$	
$x$	$c(x)$	$x$	$c(x)$
a	000	a	00
b	001	b	01
c	010	c	10
d	011	d	11
e	100	e	—
f	101	f	—
g	110	g	—
h	111	h	—





The outcomes of  $X$  ranked by their probability

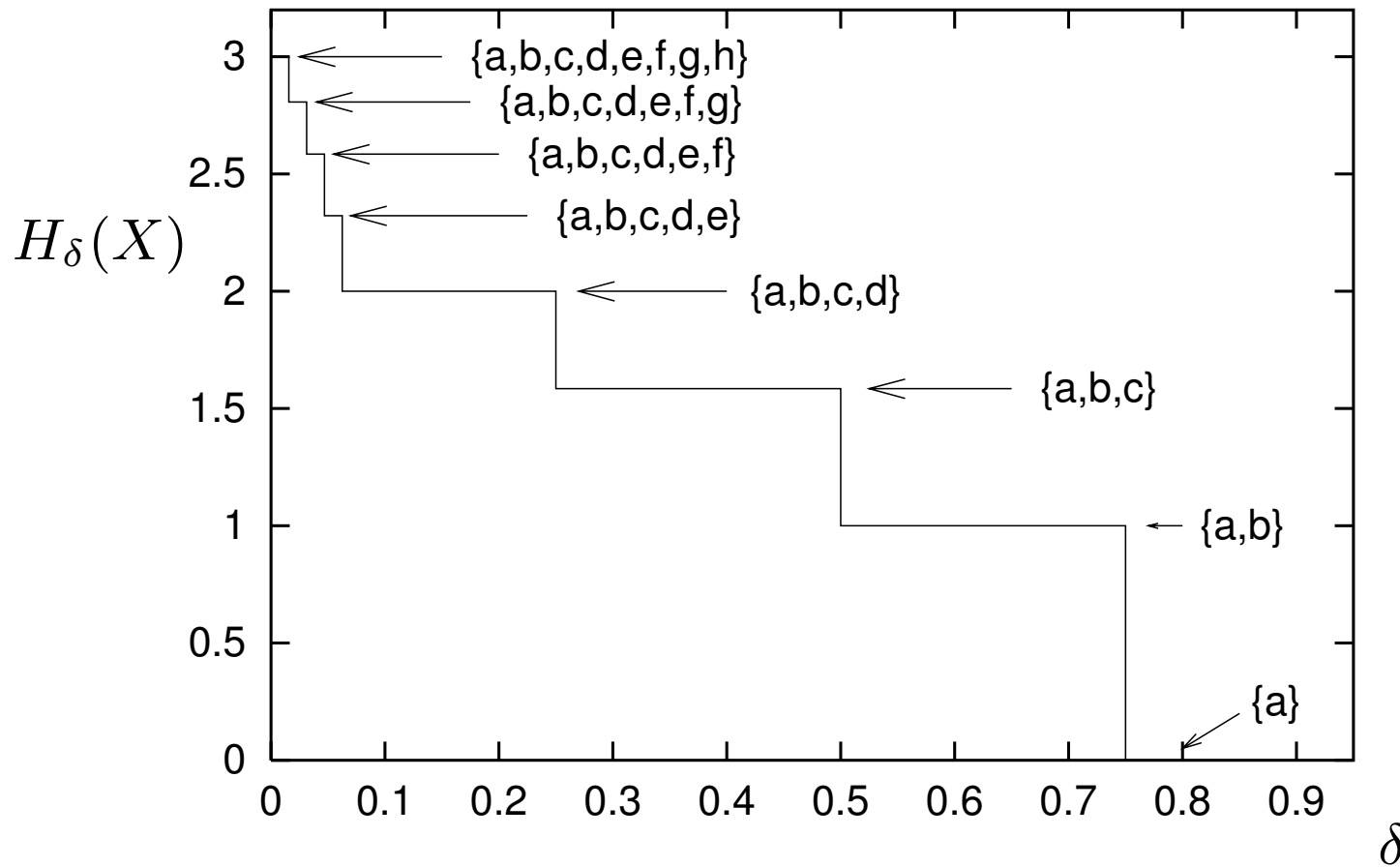
# Essential Bit Content

- Allow an error with probability  $\delta$
- Choose the smallest sufficient subset  $S_\delta$  such that  
 $P(x \in S_\delta) \geq 1 - \delta$   
(arrange the elements of  $\mathcal{A}_X$  in order of decreasing probability and take enough from beginning)

- Define: The essential bit content of  $X$  is

$$H_\delta(X) = \log_2 |S_\delta|$$

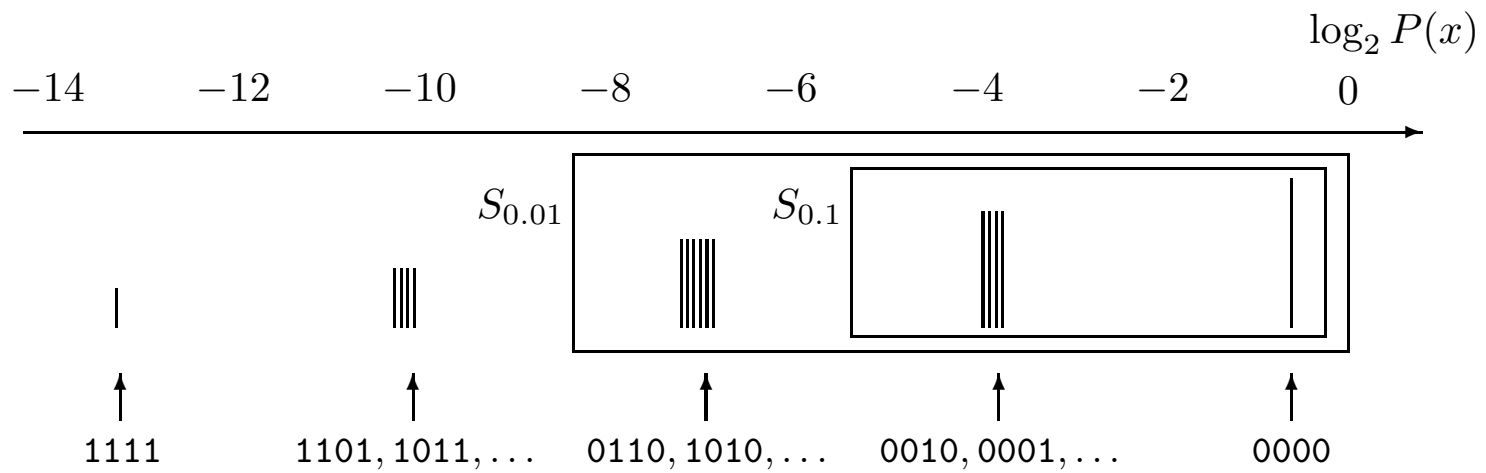
- Note that the raw bit content  $H_0$  is a special case of  $H_\delta$



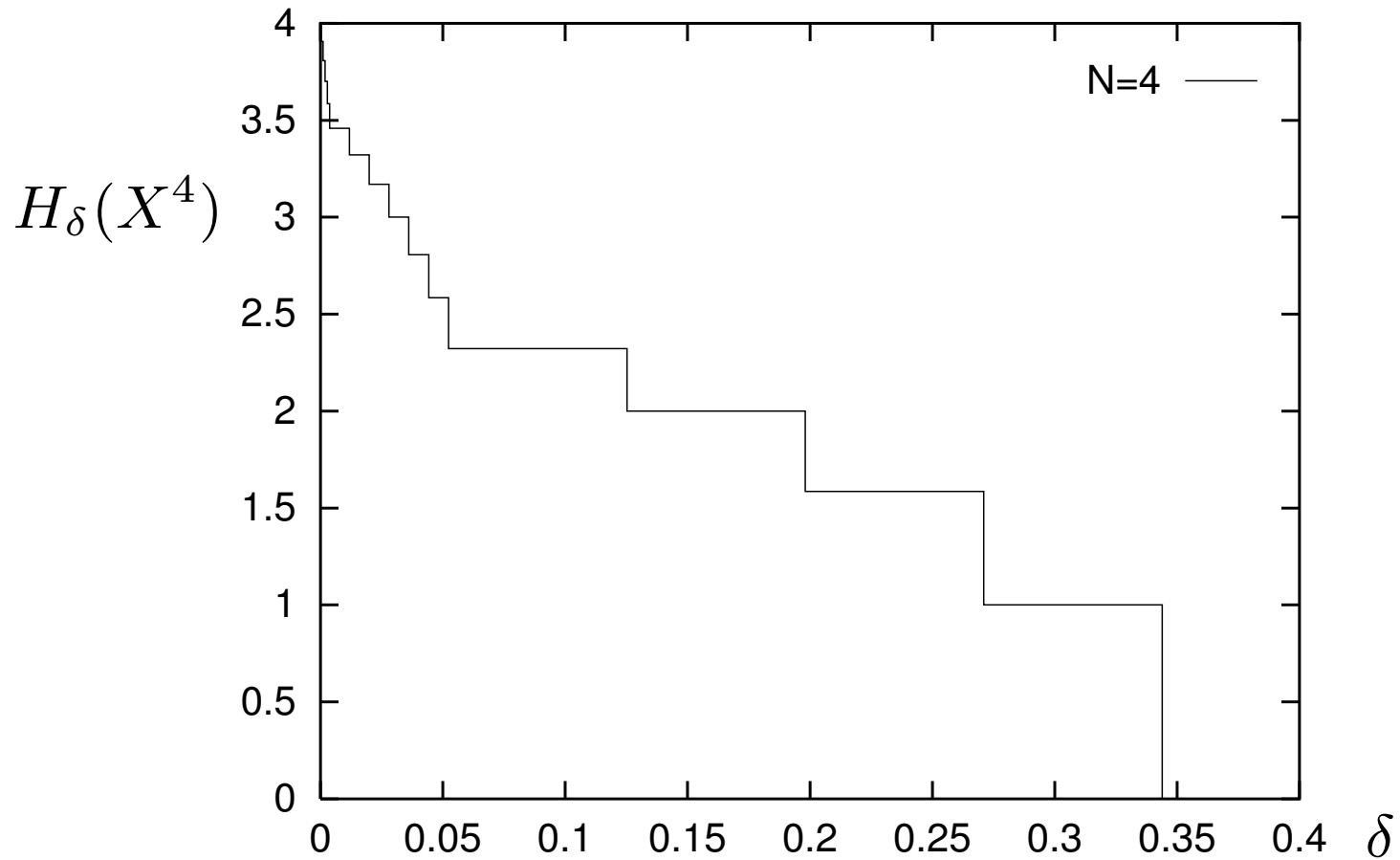
The essential bit content as the function of allowed probability of error

## Extended Ensembles (Blocks)

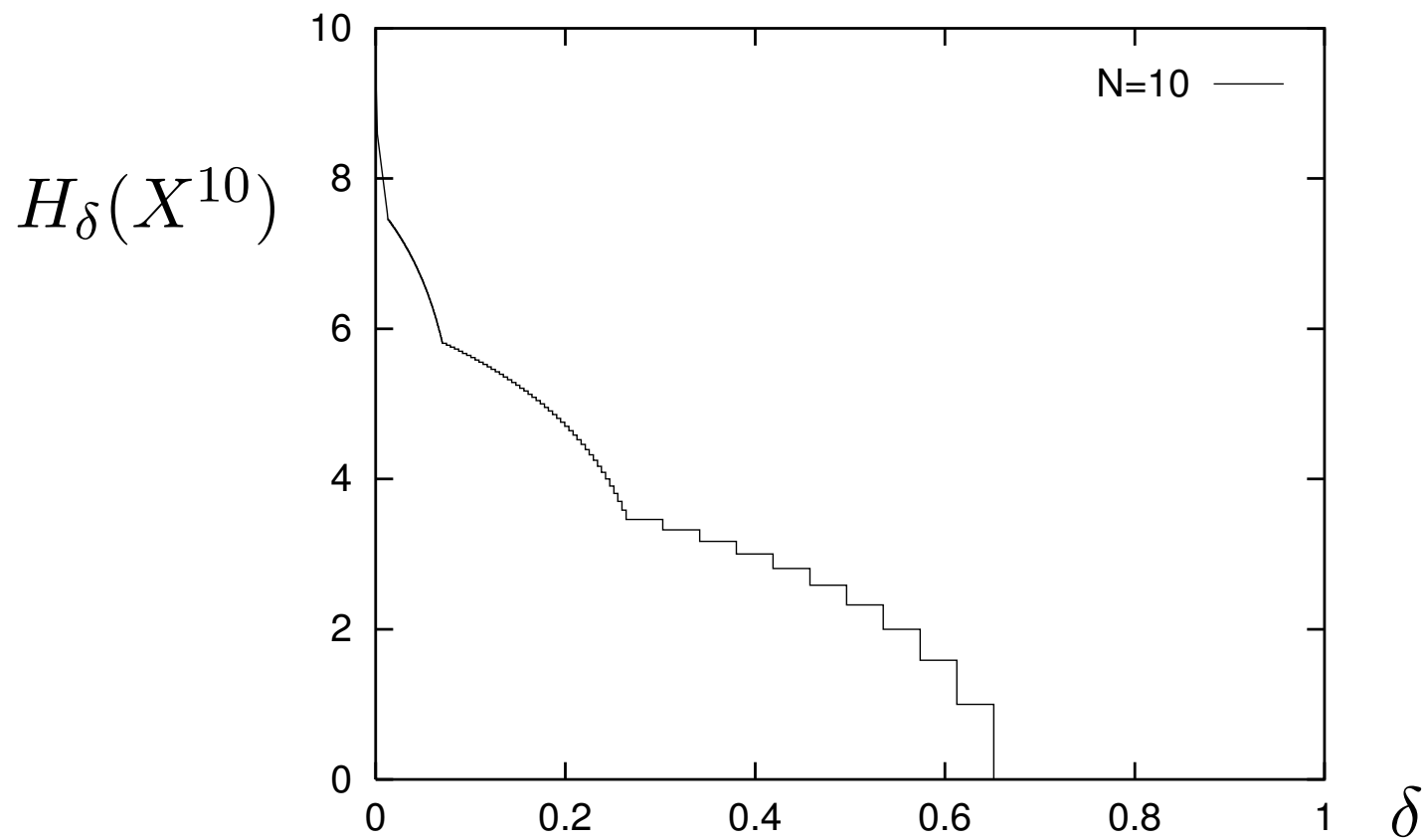
- Consider a tuple of  $N$  i.i.d. random variables
- Denote by  $X^N$  the ensemble  $(X_1, X_2, \dots, X_N)$
- Entropy is additive:  $H(X^N) = NH(X)$
- Example:  $N$  flips of a bent coin:  $p_0 = 0.9, p_1 = 0.1$



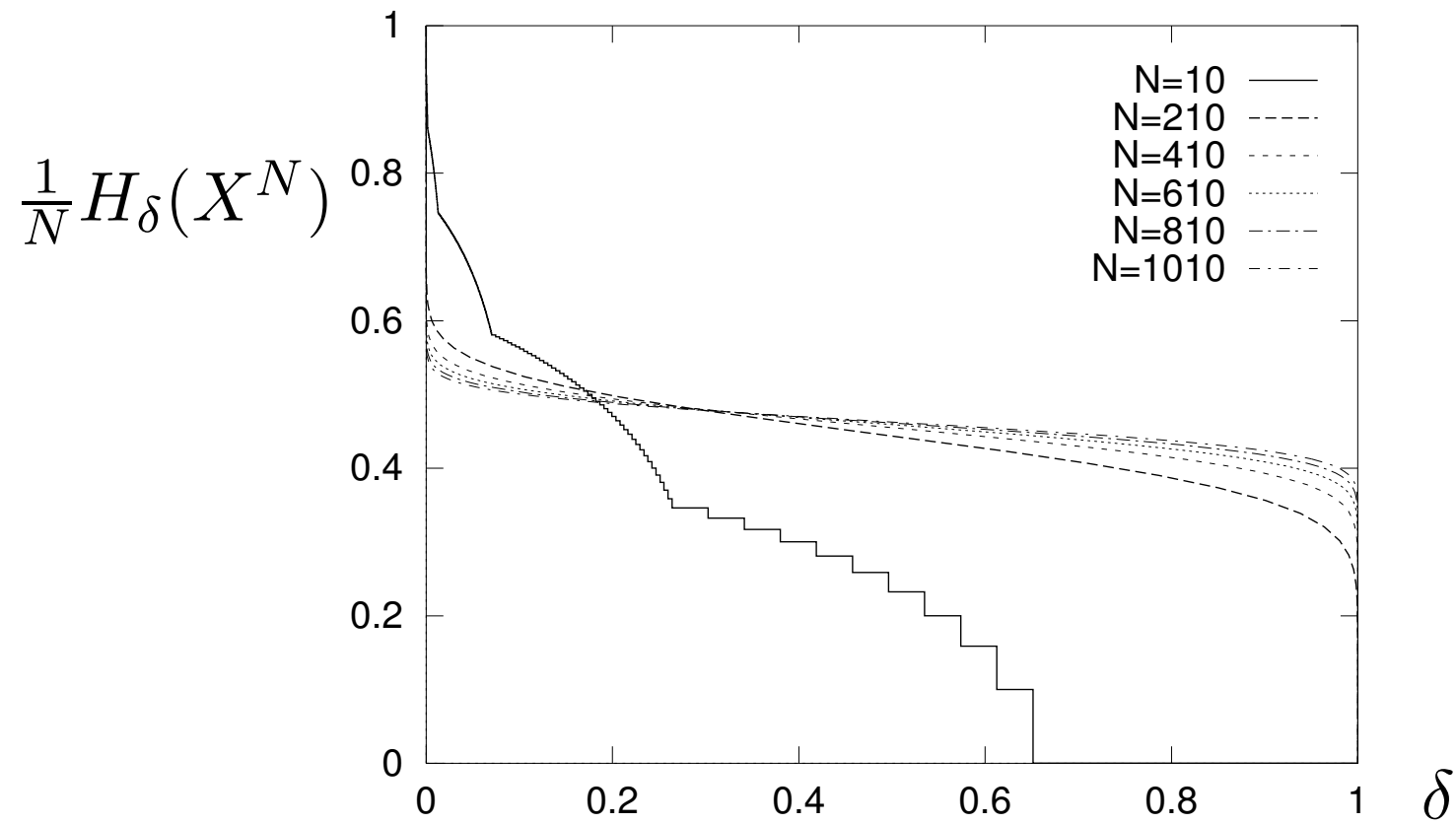
Outcomes of the bent coin ensemble  $X^4$



Essential bit content of the bent coin ensemble  $X^4$



Essential bit content of the bent coin ensemble  $X^{10}$



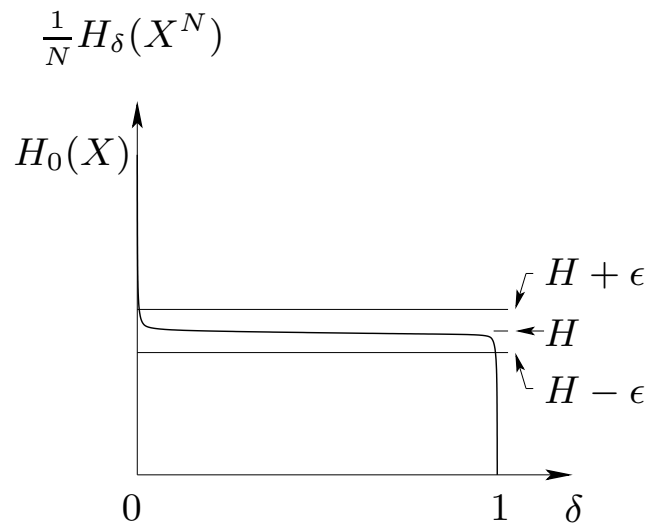
Essential bit content per toss



# Shannon's Source Coding Theorem

Given  $\epsilon > 0$  and  $0 < \delta < 1$ , there exists a positive integer  $N_0$  such that for  $N > N_0$ ,

$$\left| \frac{1}{N} H_\delta(X^N) - H(X) \right| < \epsilon.$$



- Proof involves
  - Law of large numbers
  - Chebyshev's inequality



# Typicality

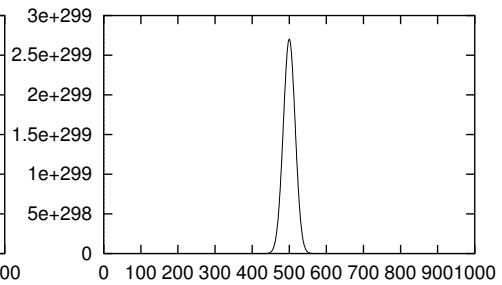
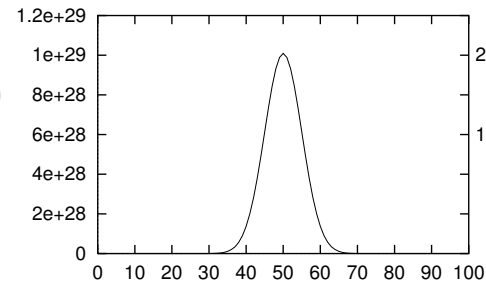
- A string contains  $r$  1s and  $N - r$  0s
- Consider  $r$  as a random variable (binomial distribution)
- Mean and std:  $r \sim Np_1 \pm \sqrt{Np_1(1 - p_1)}$
- A typical string is a one with  $r \simeq Np_1$
- In general, information content within  $N [H(X) \pm \beta]$

$$\log_2 \frac{1}{P(\mathbf{x})} = N \sum_i p_i \log_2 \frac{1}{p_i} \simeq NH(X)$$

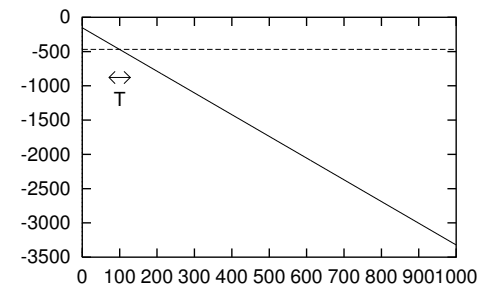
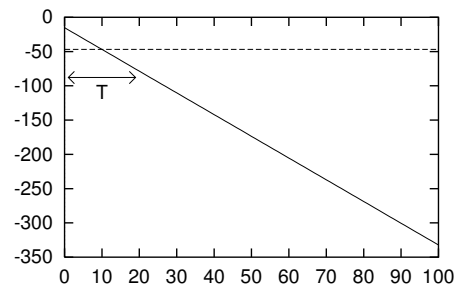
$N = 100$

$N = 1000$

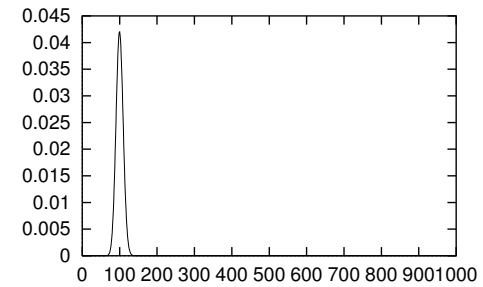
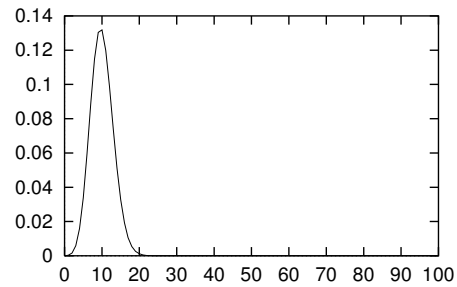
$$n(r) = \binom{N}{r}$$



$$\log_2 P(\mathbf{x})$$



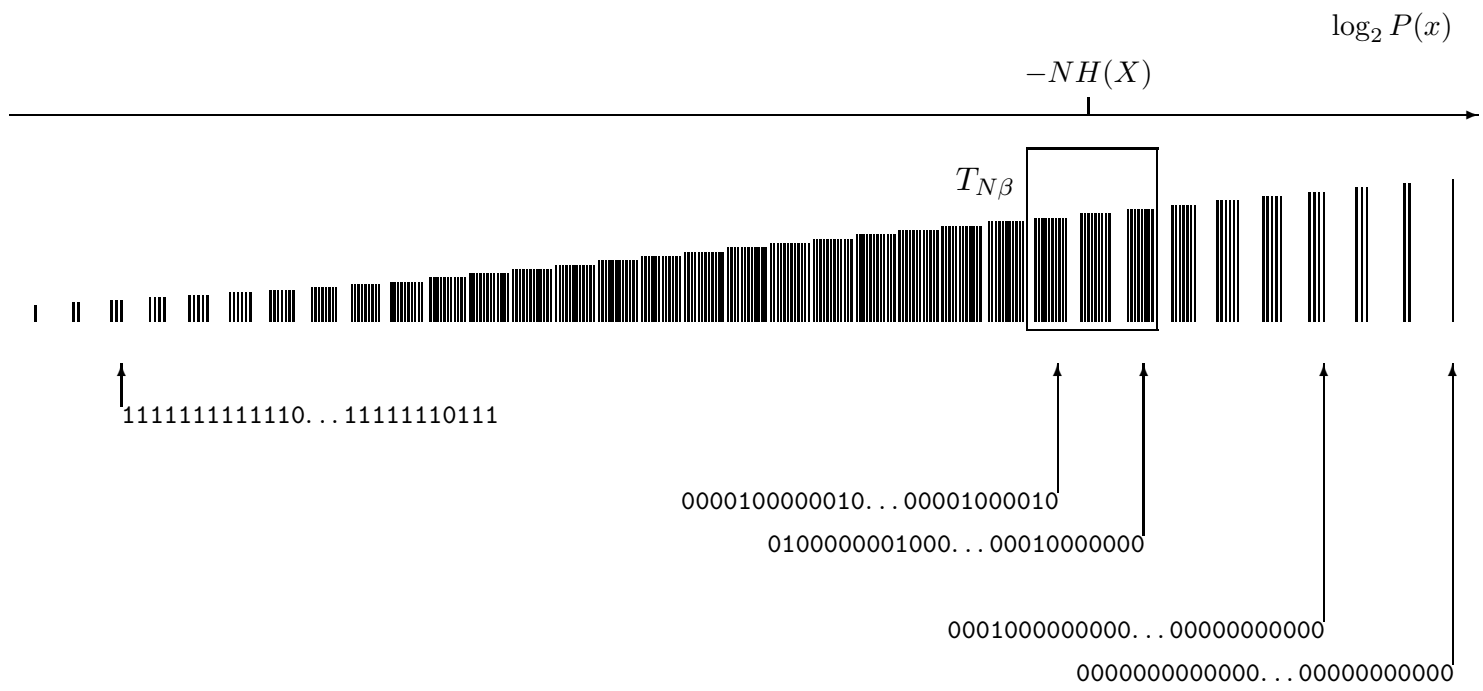
$$n(r)P(\mathbf{x}) = \binom{N}{r} p_1^r (1 - p_1)^{N-r}$$



$r$

$r$

Anatomy of the typical set  $T$



Outcomes of  $X^N$  ranked by their probability and the typical set  $T_{N\beta}$

## Shannon's source coding theorem (verbal statement)

$N$  i.i.d. random variables each with entropy  $H(X)$  can be compressed into more than  $NH(X)$  bits with negligible risk of information loss, as  $N \rightarrow \infty$ ;

conversely if they are compressed into fewer than  $NH(X)$  bits it is virtually certain that information will be lost.

## End of Chapter 4

	Chap. 4	Chap. 5	Chap. 6
Data	Block	Symbol	Stream
Lossy?	Lossy	Lossless	Lossless
Result	Shannon's source coding theorem	Huffman coding algorithm	Arithmetic coding algorithm

## Contents, Chap. 5: Symbol Codes

- Lossless coding: shorter encodings to the more probable outcomes and longer encodings to the less probable
- Practical to decode?
- Best achievable compression?
- Source coding theorem (symbol codes):  
The expected length  $L(C, X) \in [H(X), H(X) + 1)$ .
- Huffman coding algorithm



## Definitions

- A (binary) symbol code is a mapping from  $\mathcal{A}_x$  to  $\{0, 1\}^+$
- $c(x)$  is the codeword of  $x$  and  $l(x)$  its length
- Extended code  $c^+(x_1x_2 \dots x_N) = c(x_1)c(x_2) \dots c(x_N)$   
(no punctuation)
- A code  $C(X)$  is uniquely decodable if no two distinct strings have the same encoding
- A symbol code is called a prefix code if no codeword is a prefix of any other codeword (constraining to prefix codes doesn't lose any performance)

# Examples

$$\mathcal{A}_X = \{a, b, c, d\},$$

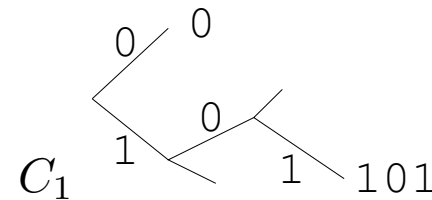
$$\mathcal{P}_X = \left\{ \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8} \right\},$$

- Using  $C_0$ :

$$c^+(\text{acdba}) = 10000010000101001000$$

- Code  $C_1 = \{0, 101\}$  is a prefix code so it can be represented as a tree
- Code  $C_2 = \{1, 101\}$  is a not prefix code because 1 is a prefix of 101

$a_i$	$c(a_i)$	$l_i$
a	1000	4
b	0100	4
c	0010	4
d	0001	4



# Expected length

- Expected length  $L(C, X)$  of a symbol code  $C$  for ensemble  $X$  is

$$L(C, X) = \sum_{x \in \mathcal{A}_X} P(x) l(x).$$

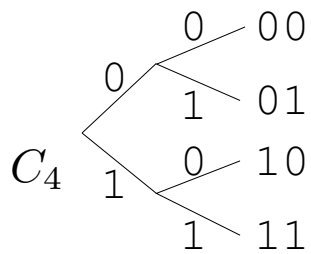
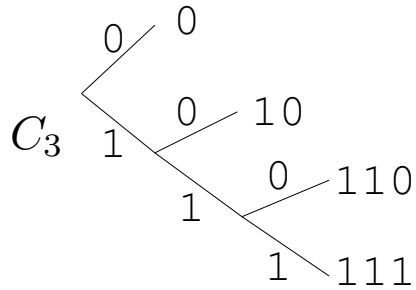
- Bounded below by  $H(X)$  (uniquely decodeable code)
- Equal to  $H(X)$  only if the codelengths are equal to the Shannon information contents:

$$l_i = \log_2(1/p_i)$$

- Codelengths implicitly define a probability distribution  $\{q_i\}$

$$q_i \equiv 2^{-l_i}$$

# Examples



- $L(C_3, X) = 1.75 = H(X)$
- $L(C_4, X) = 2 > H(X)$
- $L(C_5, X) = 1.25 < H(X)$

$C_3$ :

$a_i$	$c(a_i)$	$p_i$	$h(p_i)$	$l_i$
a	0	$1/2$	1.0	1
b	10	$1/4$	2.0	2
c	110	$1/8$	3.0	3
d	111	$1/8$	3.0	3

	$C_4$	$C_5$
a	00	0
b	01	1
c	10	00
d	11	11

## Example

$C_6$ :

$a_i$	$c(a_i)$	$p_i$	$h(p_i)$	$l_i$
a	0	$1/2$	1.0	1
b	01	$1/4$	2.0	2
c	011	$1/8$	3.0	3
d	111	$1/8$	3.0	3

- $L(C_6, X) = 1.75 = H(X)$
- $C_6$  is not a prefix code but is in fact uniquely decodable

# Kraft Inequality

- If a code is uniquely decodeable its lengths must satisfy

$$\sum_i 2^{-l_i} \leq 1$$

- For any lengths satisfying the Kraft inequality, there exists a prefix code with those lengths

0	00	000	0000	The total symbol code budget
			0001	
	001	0010		
		0011		
	01	010	0100	
			0101	
	011	0110		
		0111		
1	10	100	1000	
			1001	
	101	1010		
		1011		
	11	110	1100	
			1101	
	111	1110		
		1111		

# Source coding theorem for symbol codes

- By setting

$$l_i = \lceil \log_2(1/p_i) \rceil,$$

where  $\lceil l^* \rceil$  denotes the smallest integer greater than or equal to  $l^*$ , we get (with Kraft's inequality):

- There exists a prefix code  $C$  with

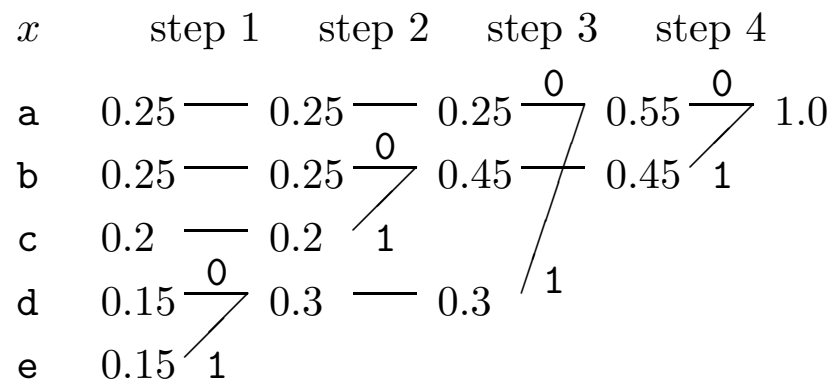
$$H(X) \leq L(C, X) < H(X) + 1$$

- Relative entropy  $D_{\text{KL}}(p||q)$  measures how many bits per symbol are wasted

$$L(C, X) = \sum_i p_i \log(1/q_i) = H(X) + D_{\text{KL}}(p||q)$$

# Huffman Coding Algorithm

1. Take two least probable symbols in the alphabet
2. Give them the longest codewords differing only in the last digit
3. Combine them into a single symbol and repeat



$a_i$	$p_i$	$h(p_i)$	$l_i$	$c(a_i)$
a	0.25	2.0	2	00
b	0.25	2.0	2	10
c	0.2	2.3	2	11
d	0.15	2.7	3	010
e	0.15	2.7	3	011



# Optimality

- Huffman coding is optimal in two senses:
  - Smallest expected codelength of uniquely decodable symbol codes
  - Prefix code → easy to decode
- But:
  - The overhead of between 0 and 1 bits per symbol is important if  $H(X)$  is small → compress blocks of symbols to make  $H(X)$  larger
  - Does not take context into account (symbol code vs. stream code)

## End of Chapter 5

	Chap. 4	Chap. 5	Chap. 6
Data	Block	Symbol	Stream
Lossy?	Lossy	Lossless	Lossless
Result	Shannon's source coding theorem	Huffman coding algorithm	Arithmetic coding algorithm

# Guessing Game

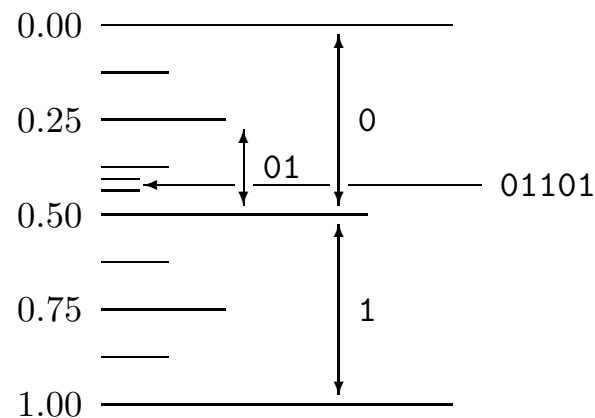
- Human was asked to guess a sentence character by character
- The numbers of guesses are listed below each character

T H E R E - I S - N O - R E V E R S E - O N - A - M O T O R C Y C L E -  
1 1 1 5 1 1 2 1 1 2 1 1 5 1 1 7 1 1 1 2 1 3 2 1 2 2 7 1 1 1 1 4 1 1 1 1 1

- One could encode only the string 1, 1, 1, 5, 1, ...
- Decoding requires an identical twin who also plays the guessing game

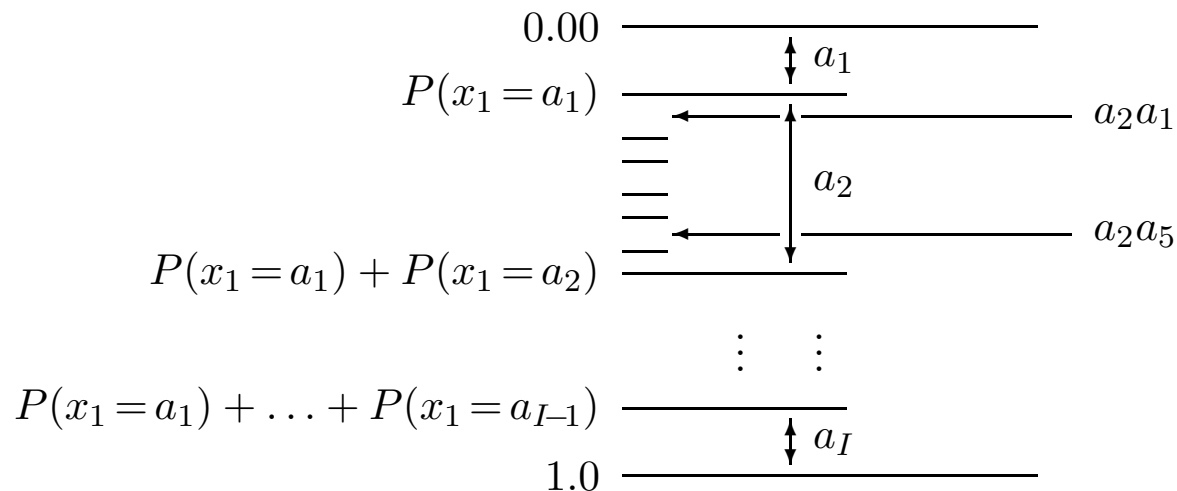
## Arithmetic Coding (1/2)

- Human predictor is replaced by a probabilistic model of the source
- The model supplies a predictive distribution over the next symbol
- It can handle complex adaptive models (context-dependent)
- Binary strings define real intervals within the real line  $[0, 1)$
- The string 01 corresponds to  $[0.01, 0.10)$  in binary or  $[0.25, 0.50)$  in base ten



## Arithmetic Coding (2/2)

- Divide the real line  $[0, 1)$  into  $I$  intervals of lengths equal to the probabilities  $P(x_1 = a_i)$



- Pick an interval and subdivide it (and iterate)
- Send a binary string whose interval lies within that interval

## Example: Bent Coin (1/3)

- Coin sides are a and b, and the 'end of file' symbol is  $\square$
- Use a Bayesian model with a uniform prior over probabilities of outcomes

---

Context (sequence thus far)	Probability of next symbol		
	$P(a) = 0.425$	$P(b) = 0.425$	$P(\square) = 0.15$
b	$P(a   b) = 0.28$	$P(b   b) = 0.57$	$P(\square   b) = 0.15$
bb	$P(a   bb) = 0.21$	$P(b   bb) = 0.64$	$P(\square   bb) = 0.15$
bbb	$P(a   bbb) = 0.17$	$P(b   bbb) = 0.68$	$P(\square   bbb) = 0.15$
bbba	$P(a   bbba) = 0.28$	$P(b   bbba) = 0.57$	$P(\square   bbba) = 0.15$

---

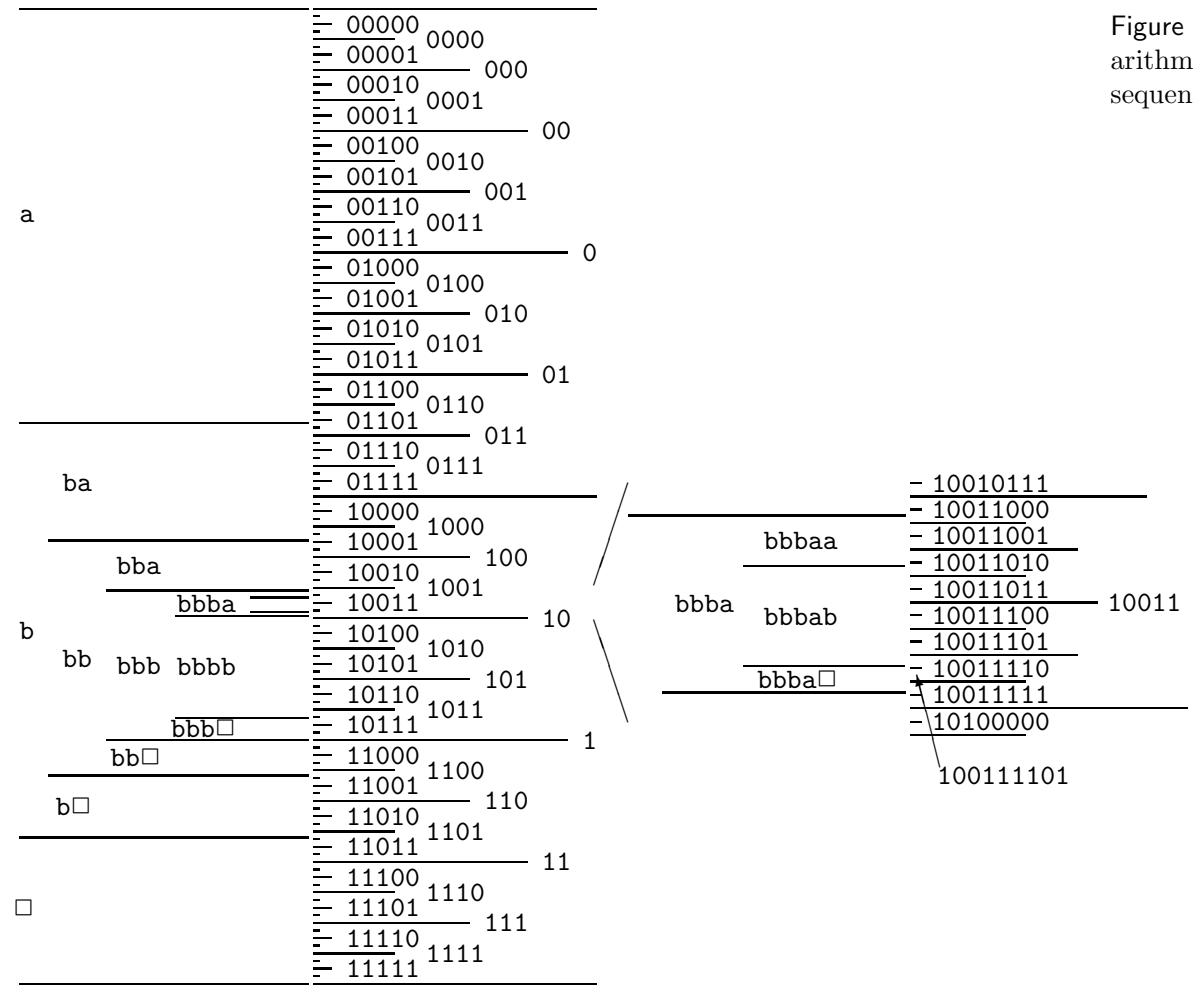


Figure 6.4. Illustration of the arithmetic coding process as the sequence **bbba**□ is transmitted.

		00000	
		00001	0000
	aaaa	00010	000
	aaa	00011	0001
		00100	00
	aa	00101	0010
		00110	001
	aab	00111	0011
		01000	0
	aba	01001	0100
		01010	010
	abb	01011	0101
		01100	01
	a□	01101	0110
		01110	011
	baa	01111	0111
		10000	
	ba	10001	1000
		10010	100
	bab	10011	1001
		10100	10
	b	10101	1010
		10110	101
	bb	10111	1011
		11000	1
	bb□	11001	1100
		11010	110
	b□	11011	1101
		11100	11
		11101	1110
	□	11110	111
		11111	1111



# On Arithmetic Coding

- Computationally efficient
- Length of a string closely matches the Shannon information content
- Overhead required to terminate a message is never more than 2 bits  
⇒ Finding a good coding equivalent to finding a good probabilistic model!
- Flexible:
  - any source alphabet and any encoded alphabet
  - alphabets can change with time
  - probabilities are context-dependent
- Can be used to generate random samples from random bits economically

# Lempel-Ziv Coding

- Used in gzip etc.

source substrings	$\lambda$	1	0	11	01	010	00	10
$s(n)$	0	1	2	3	4	5	6	7
$s(n)_{\text{binary}}$	000	001	010	011	100	101	110	111
(pointer, bit)		(, 1)	(0, 0)	(01, 1)	(10, 1)	(100, 0)	(010, 0)	(001, 0)

- Asymptotically compress down to the entropy of the source  
(not in practice)

## Summary (1/2)

- Fixed-length block codes (Chapter 4)
  - Only a tiny fraction of source strings are given an encoding
  - Identify entropy as the measure of compressibility
  - No practical use
- Symbol codes (Chapter 5)
  - Variable code lengths allow lossless compression
  - Expected code length is  $H + D_{KL}$  (between the source distribution and the code's implicit distribution)
  - $D_{KL}$  can be made smaller than 1 bit per symbol
  - Huffman code is the optimal symbol code

## Summary (2/2)

- Stream codes (Chapter 6)
  - Arithmetic coding combines a probabilistic model with an encoding algorithm
  - Lempel-Ziv memorises strings that have already occurred
  - If any of the bits is altered by noise, the rest of the encoding fails

# Exercises

- 6.19 (entropy and information)
- 4.16 (Shannon source coding theorem)
- 6.16 (Huffman coding)
- 6.7 (Arithmetic coding)