Lossless Source Coding

Gadiel Seroussi *Marcelo Weinberger*

Information Theory Research
Hewlett-Packard Laboratories
Palo Alto, California

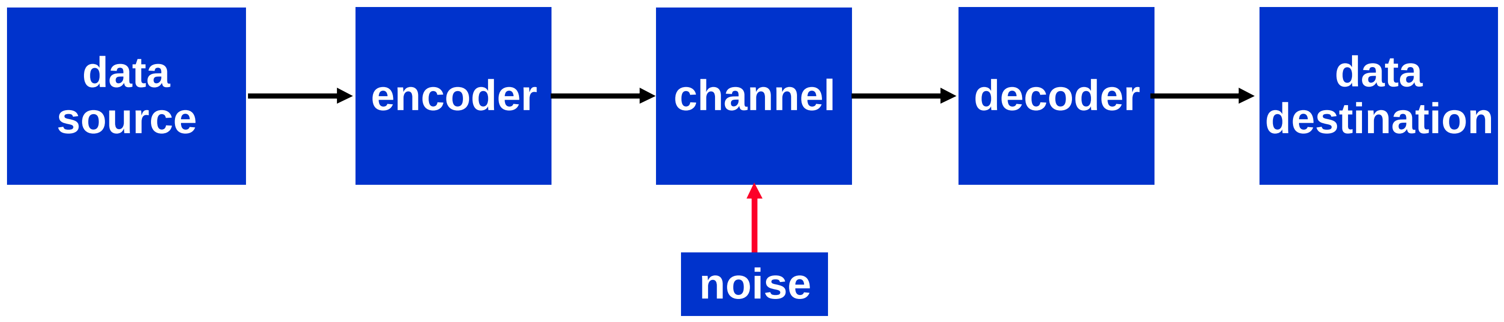
Lossless Source Coding

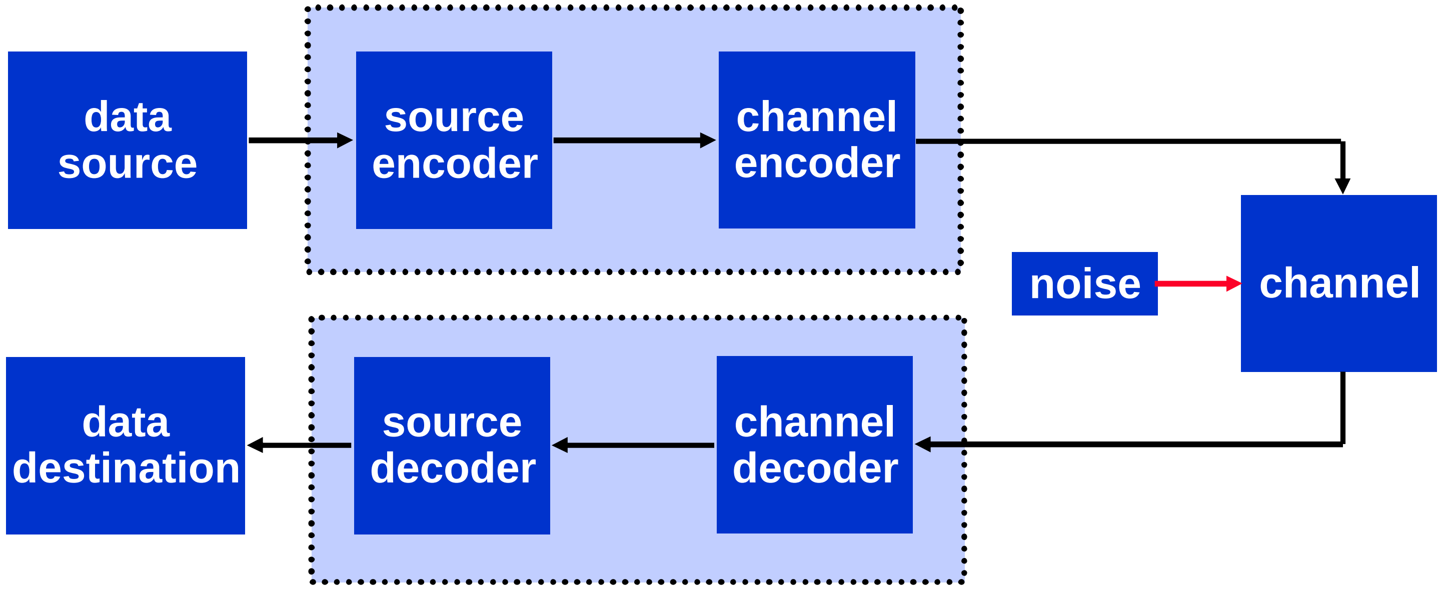
1. Fundamentals

Notation

- A = discrete (usually finite) alphabet
- $\alpha = |A| =$ size of A (when finite)
- $x_1^n = x^n = x_1 x_2 x_3 \ldots x_n =$ finite sequence over A
- $x_1^\infty = x^\infty = x_1 x_2 x_3 \ldots x_t \ldots =$ infinite sequence over A
- $x_i^j = x_i x_{i+1} \ldots x_j =$ sub-sequence (i sometimes omitted if $= 1$)
- $p_X(x) = \mathrm{Prob}(X{=}x) =$ probability mass function (PMF) on A
  (subscript X and argument x dropped if clear from context)
- $X \sim p(x)$: X obeys PMF $p(x)$
- $E_p[F] =$ expectation of F w.r.t. PMF p (subscript and $[\ ]$ may be dropped)
- $\hat{p}_{x_1^n}(x) =$ empirical distribution obtained from $x_1{}^n$
- $\log x =$ logarithm to base 2 of x, unless base otherwise specified
- $\ln x =$ natural logarithm of x
- $H(X), H(p) =$ entropy of a random variable X or PMF p, in bits; also
- $H(p) = -p \log p - (1-p)\log(1-p),\ 0 \le p \le 1$: binary entropy function
- $D(p||q) =$ relative entropy (information divergence) between PMFs p and q

Coding in a communication/storage system

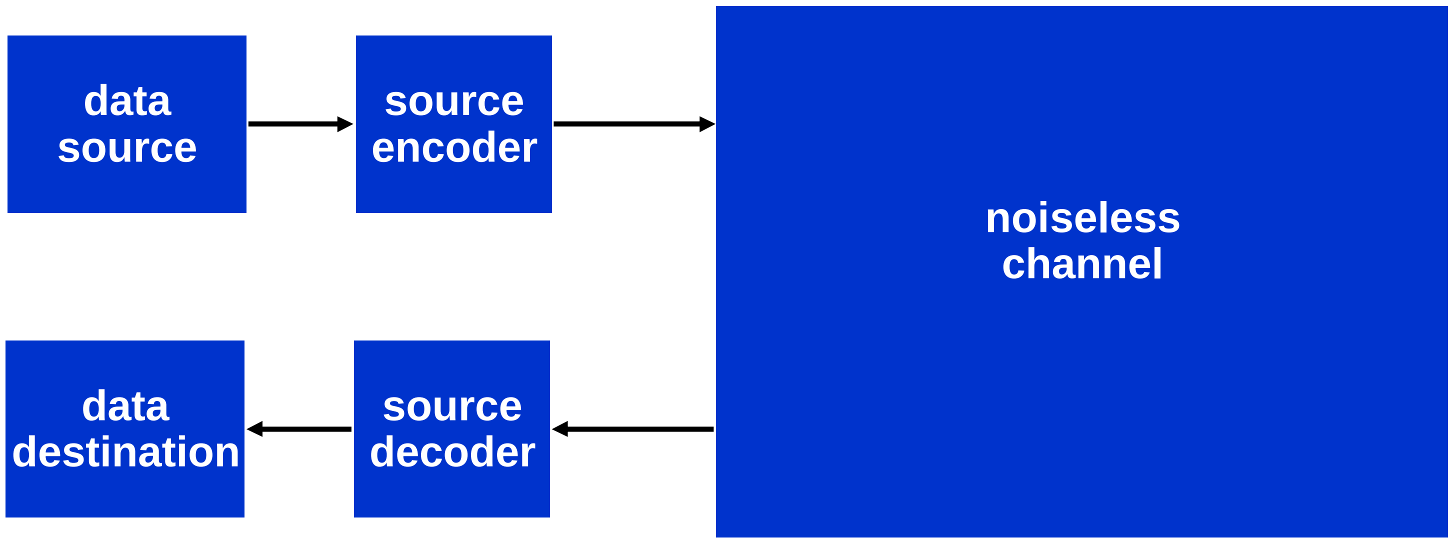




Information Theory

- **Shannon, *“A mathematical theory of communication,” Bell Tech. Journal,* 1948**
  - **Theoretical foundations of source and channel coding**
  - **Fundamental bounds and coding theorems in a probabilistic setting**
    - in a nutshell: perfect communication in the presence of noise is possible as long as the *entropy rate* of the source is below the *channel capacity*
  - **Fundamental theorems essentially non-constructive: we’ve spent the last 52 years realizing Shannon’s promised paradise in practice**
    - very successful: enabled current digital revolution (multimedia, internet, wireless communication, mass storage, …)
  - **Separation theorem: *source and channel coding can be done independently***

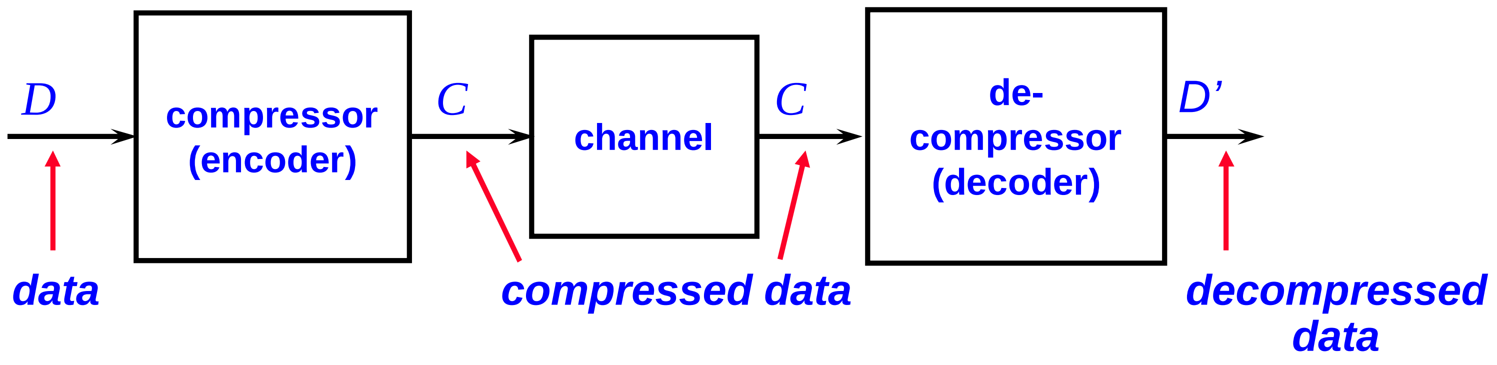
Source Coding



Source coding = Data compression

Þ efficient use of bandwidth/space

Data Compression

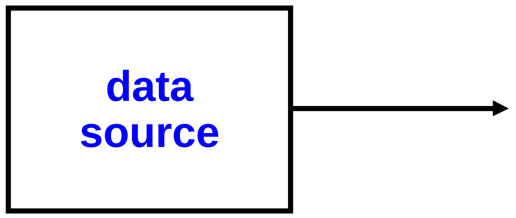


the goal: $size(C) < size(D)$

compression ratio: $r = \frac{size(C)}{size(D)} < 1$ **in appropriate units, e.g., bits/symbol**

- **Channel:**
  - **Communications channel ("from here to there")**
  - **Storage channel ("from now to then")**
- ***Lossless* compression:** $D = D'$ **the case of interest here**
- ***Lossy* compression:** D' **is an approximation of** D **under some metric**

Data Sources



$x_1 x_2 x_3 \ldots x_t \ldots x_n \triangleq x_1^n$

- Symbols $x_i \in A =$ a ***countable*** (usually ***finite***) ***alphabet***
- ***Probabilistic source:*** x_i **are** ***random variables;*** x_1^n **obeys some probability distribution** P **on** A^n **(the** ***ensemble*** **of possible sequences emitted by the source)**
  - **we are often interested in** $n \to \infty$ **:** x_1^∞ **is a** ***random process***
    - *stationary* (*time-invariant*): $x_i^\infty = x_j^\infty$, as random processes, $\forall \; i,j \geq 1$
    - *ergodic*: time averages converge (to ensemble averages)
    - *memoryless*: x_i are statistically independent
    - independent, identically distributed (*i.i.d.)*: memoryless, and $x_i \sim p_X \;\; \forall \; i$
- ***Individual sequence:*** x_i **are just symbols, not assumed to be a realization of a random process. The "data source" will include a probability assignment, but it will be derived from the data under certain constraints, and with certain objectives**

Statistics on Individual Sequences

- **Empirical distributions**

$$\hat{p}_{x_1^n}(a) = \frac{1}{n}\left|\{x_i \mid 1 \le i \le n,\, x_i = a\}\right|, \quad a \in A$$

memoryless, Bernoulli model

  - **we can compute empirical statistics of *any order* (joint, conditional, etc.)**
  - **sequence probability according to own empirical distribution**

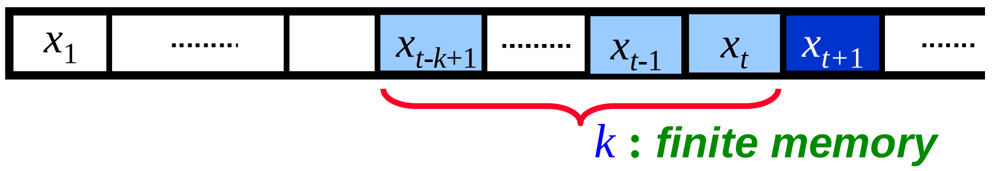
$$\hat{P}_{x_1^n}(x_1^n) = \prod_{i=1}^{n} \hat{p}_{x_1^n}(x_i)$$

  - **this is the *highest probability* assigned to the sequence by any distribution from the model class (*maximum likelihood estimator*)**
  - **Example:** $A = \{0,1\},\ \ n_0 = \left|\{i \mid x_i = 0\}\right|,\ \ n_1 = n - n_0 = \left|\{i \mid x_i = 1\}\right|$

$$\hat{p}(0) = \frac{n_0}{n},\ \ \hat{p}(1) = \frac{n_1}{n}\ :\qquad \hat{P}(x_1^n) = \hat{p}(0)^{n_0}\,\hat{p}(1)^{n_1} = \frac{n_0^{n_0} n_1^{n_1}}{n^n}$$

  - **Notice that if x_1^n is in fact the outcome of a random process, then its empirical statistics are themselves random variables**
    - **e.g., expressions of the type** $\Pr_p(|\,\hat{p}(a) - p(a)\,| \ge \epsilon)$

Statistical Models for Data Sources



q ***Markov*** **of order** k ³ 0

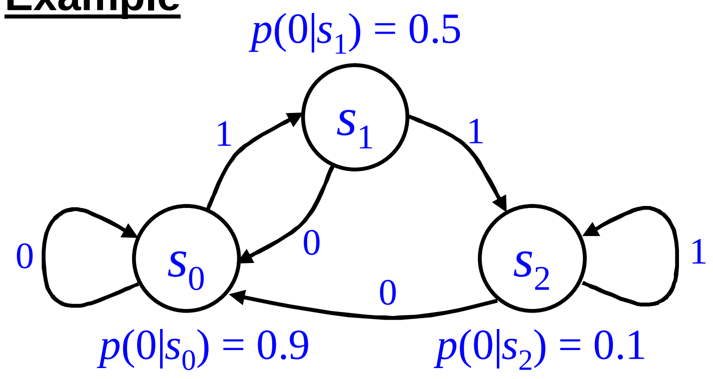
$$p(x_{t+1} \mid x_1^t) = p(x_{t+1} \mid x_{t-k+1}^t), \quad t \geq k$$ **(some convention for** $t < k$**)**

- l **i.i.d = Markov of order 0**

q ***Finite State Machine (FSM)***

- l ***state space*** $S = \{s_0, s_1, \ldots, s_{K-1}\}$
- l ***initial state*** s_0
- l ***transition probability***
  $q(s \mid s', a), \quad s, s' \in S,\ a \in A$
- l ***output probability*** $p(a \mid s),\ \ a \in A,\ s \in S$
- l ***unifilar*** Û **deterministic transitions:**
  next-state function $f : S$ ´ $A \rightarrow S$
- l **every Markov source is equivalent to a unifilar FSM with** $K \leq |A|^k$**, but in general, finite state** ¹ **finite memory**

Example



Steady state

state probs.

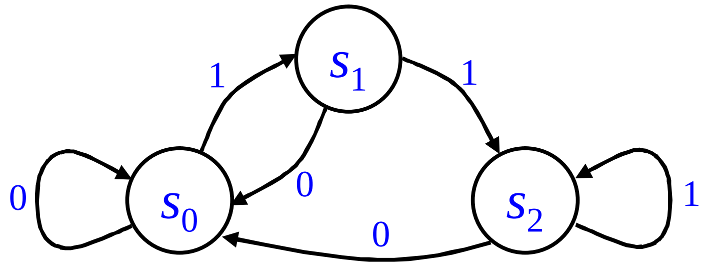
$$[p_0\, p_1\, p_2] \begin{bmatrix} .9 & .1 & 0 \\ .5 & 0 & .5 \\ .1 & 0 & .9 \end{bmatrix} = [p_0\, p_1\, p_2]$$

symb. probs.

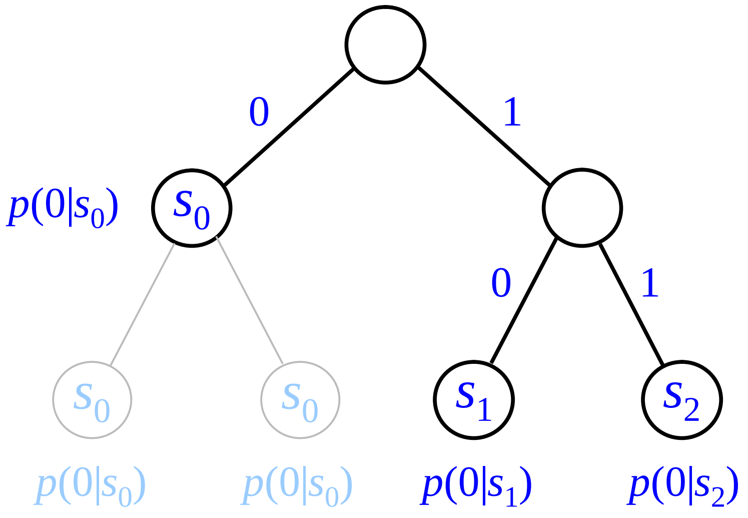
$$\Rightarrow [p_0\, p_1\, p_2] = \begin{bmatrix} \frac{5}{8} & \frac{1}{16} & \frac{5}{16} \end{bmatrix}, \quad [p_0\ \ p_1] = \begin{bmatrix} \frac{5}{8} & \frac{3}{8} \end{bmatrix}$$

Statistical Models for Data Sources (cont.)

- ***Tree sources (FSMX)***
  - finite memory $\leq k$ (Markov)
  - # of past symbols needed to determine the state might be $< k$ for some states



$$s_0 \leftrightarrow 0, \quad s_1 \leftrightarrow 01, \quad s_2 \leftrightarrow 11$$



- by merging nodes from the full Markov tree, we get a model with a ***smaller number of free parameters***
- the set of tree sources with unbalanced trees has ***measure zero*** in the space of Markov sources of any given order
- yet, tree source models have proven very useful in practice, and are associated with some of the best compression algorithms to date
- more about this later ...

Entropy

$$\boxed{X \sim p(x) \;:\; H(X) = -\sum_{x\in A} p(x)\log p(x)} \qquad [0\log 0 \overset{\Delta}{=} 0]$$

entropy **of** X **(or of the PMF** $p(\cdot)$ **), measured in *bits***

$$H(X) = E_p[-\log p(X)]$$

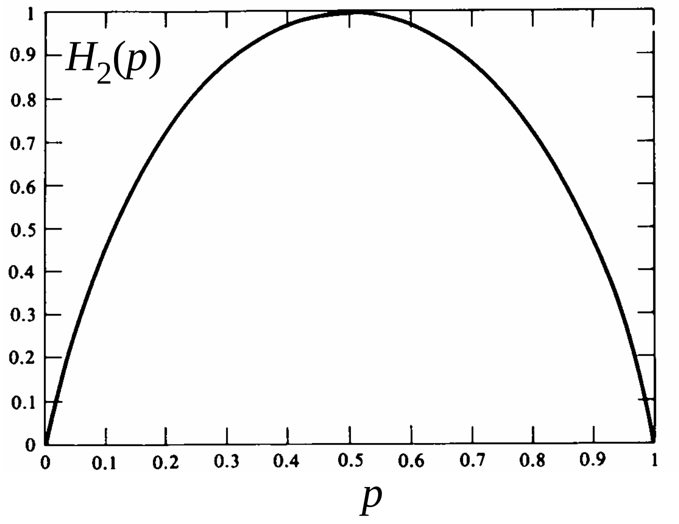
- H **measures the *uncertainty* or *self-information* of** X
- **we also write** $H(p)$ **: a random variable is not actually needed;** $p(\cdot)$ **could be an empirical distribution**

Example: $A=\{0,1\}$, $p_X(1) = p$, $p_X(0) = 1-p$ **(overloaded notation!)**

$$\boxed{H_2(p) = -p\log p - (1-p)\log(1-p)}$$ ***binary entropy function***

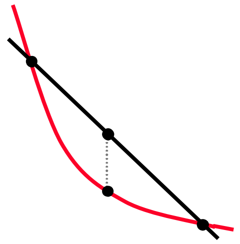
Main properties:

- $H_2(p) \geq 0$, $H_2(p)$ **is** $\cap$**-convex,** $0 \leq p \leq 1$
- $H_2(p) \to 0$ **as** $p \to 0$ **or** 1**, with slope** ∞
- $H_2(p)$ **is maximal at** $p = 0.5$**,** $H_2(0.5)=1$
  - Þ **the entropy of an unbiased coin is 1 bit**



Entropy (cont.)

- For a general finite alphabet A, $H(X)$ is maximal when $X \sim p_u$, where $p_u(a)=1/|A|$ for all $a\in A$ (uniform distribution)
  - **Jensen's inequality:** if f is a È-convex function, then $Ef(X) \geq f(EX)$.
  - $-\log x$ is a È-convex function of x
  - $H(X) = E[\log(1/p(X))] = -E[-\log(1/p(X))] \leq \log E[1/p(X)] = \log|A| = H(p_u)$



- **Empirical entropy**
  - entropy computed an an empirical distribution
  - **Example:** recall that the probability assigned to an individual binary sequence by its own zero-order empirical distribution is

$$\hat{P}(x_1^n) = \hat{p}(0)^{n_0}\,\hat{p}(1)^{n_1} = \frac{n_0^{n_0} n_1^{n_1}}{n^n}$$

  - we have $$-\frac{1}{n}\log\hat{P}(x_1^n) = -\frac{n_0}{n}\log\left(\frac{n_0}{n}\right) - \frac{n_1}{n}\log\left(\frac{n_1}{n}\right) = H_2(\hat{p}(0)) = \hat{H}(x_1^n)$$

normalized, in bits/symbol

in fact, $-\frac{1}{n}\log\hat{P}(x_1^n) = \hat{H}(x_1^n)$ holds for a large class of probability models

Joint and Conditional Entropies

- The ***joint entropy*** of random variables $(X,Y) \sim p(x,y)$ is defined as

$$\mathbf{H}(X,Y) = -\sum_{x,y} p(x,y) \log p(x,y)$$

  - this can be extended to any number of random variables: $\mathbf{H}(X_1,X_2,\ldots, X_n)$

Notation: $\mathbf{H}(X_1,X_2,\ldots, X_n) =$ joint entropy of $X_1,X_2,\ldots, X_n$ $\quad (0 \le \mathbf{H} \le n \log |A|)$

$H(X_1,X_2,\ldots, X_n) = \mathbf{H}/n =$ ***normalized per-symbol entropy*** $\quad (0 \le H \le \log |A|)$

  - if (X,Y) are statistically independent, then $\mathbf{H}(X,Y) = H(X) + H(Y)$

- The ***conditional entropy*** is defined as

$$H(Y \mid X) = \sum_x p(x) H(Y \mid X = x) = -E_{p(x,y)} \log p(Y \mid X)$$

- ***Chain rule***:

$$\mathbf{H}(X,Y) = H(X) + H(Y \mid X)$$

- ***Conditioning reduces uncertainty (on the average)***:

$$H(X \mid Y) \le H(X)$$

  - but $H(X|Y{=}y)$ ³ $H(X)$ is possible

Entropy Rates

- *Entropy rate of a random process*

$$\boxed{H(X_1^\infty) = \lim_{n\to\infty} \frac{1}{n}\mathbf{H}(X_1^n)}$$

in bits/symbol,
if the limit exists!

- **A related limit based on *conditional entropy***

$$\boxed{H^*(X_1^\infty) = \lim_{n\to\infty} H(X_n | X_{n-1}, X_{n-2}, \ldots, X_1)}$$

in bits/symbol,
if the limit exists!

***Theorem*: For a stationary random process, both limits exist, and**

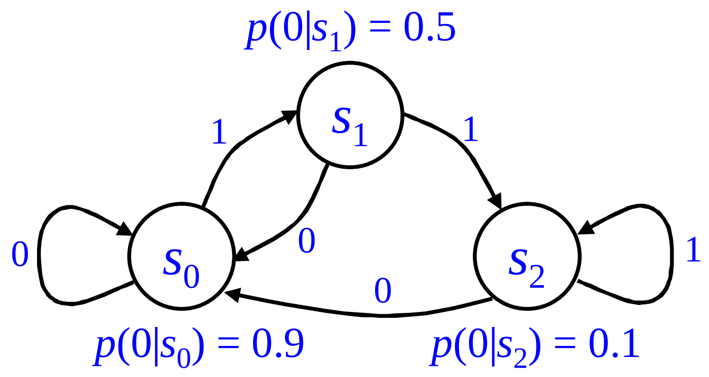
$$H^*(X_1^\infty) = H(X_1^\infty)$$

- **Examples:**
  - $X_1, X_2, \ldots$ **i.i.d.:** $H(X_1^\infty) = \lim_{n\to\infty} \mathbf{H}(X_1, X_2, \ldots, X_n)/n = \lim_{n\to\infty} nH(X_1)/n = H(X_1)$
  - X_1^∞ **stationary** k**-th order Markov:**

$$H(X_1^\infty) \overset{\text{theorem}}{=} H^*(X_1^\infty) = \lim_{n\to\infty} H(X_n | X_{n-1}, \ldots, X_1) \overset{\text{Markov}}{=} \lim_{n\to\infty} H(X_n | X_{n-1}, \ldots, X_{n-k}) \overset{\text{stationary}}{=} H(X_{k+1} | X_k, \ldots, X_1)$$

The theorem provides a very useful tool to compute entropy rates for a broad family of source models

Entropy Rates - Examples



Steady state

$$[p_0\, p_1\, p_2] = \left[\frac{5}{8}\ \ \frac{1}{16}\ \ \frac{5}{16}\right], \quad [p_0\ \ p_1] = \left[\frac{5}{8}\ \ \frac{3}{8}\right]$$

state probs. symb. probs.

- **Zero-order entropy**

$$H(0.375) = 0.954$$

- **Markov process entropy**

$$H(X\,|\,S) = \sum_{i=0}^{2} p(s_i) H(p(0\,|\,s_i)) =$$

$$\frac{5}{8} H(0.9) + \frac{1}{16} H(0.5) + \frac{5}{16} H(0.1) \approx 0.502$$

- **Individual sequence - fitted with FSM model**

0000001111111111100000100000000111111110 s_0 s_1 s_2

Empirical entropy:

$$\hat{p}(0\,|\,s_0) = \frac{16}{19},\ \hat{p}(0\,|\,s_1) = \frac{1}{3},\ \hat{p}(0\,|\,s_2) = \frac{1}{9}, \quad [\hat{p}_0\ \hat{p}_1\ \hat{p}_2] = \left[\frac{19}{40}\ \frac{3}{40}\ \frac{18}{40}\right], \quad \hat{H}(x\,|\,S) = 0.594$$

Relative Entropy

q **The *relative entropy* (or *Kullback-Leibler distance*, or *information divergence*) between two PMFs $p(x)$ and $q(x)$ is defined as**

$$D(p\,\|\,q)=\sum_x p(x)\log\frac{p(x)}{q(x)}=E_p\,\frac{p(x)}{q(x)}$$

Theorem: $D(p\|q)$ ³ 0, **with equality iff** $p=q$

- u Proof (using strict concavity of $\log$, and Jensen's inequality):

$$-D(p\,\|\,q)=\sum_x p(x)\log\frac{q(x)}{p(x)}\leq\log\sum_x p(x)\frac{q(x)}{p(x)}=\log\sum_x q(x)\leq 0$$

the summations are over values of x where $p(x)\,q(x)\neq 0$; other terms contribute either 0 or ∞ to D. Since $\log$ is strictly concave, equality holds iff $p(x)/q(x)=1\ \forall x$. g

- l **D is not symmetric, and therefore not a distance in the metric sense**
- l **however, it is a very useful way to express 'proximity' of distributions**

in a sense, $D(p\|q)$ measures the inefficiency of assuming that the distribution is q when it is actually p

Maximal Entropy

Theorem: ***Let*** $X \sim p$ ***be a PMF over*** $\mathbb{Z}_{>0}$ ***such that*** $E_p X = \mu$ **. Then** $H(X)$ ***is maximized when*** $p(x) = \exp(\lambda_0 + \lambda_1 x)$ ***satisfying the constraint***

- ***a similar theorem holds for moments of any order***
- **Proof: Consider a PMF** q **satisfying the constraint. Then show** $H(q) \le H(p)$ **using non-negativity of** $D(q\|p)$**,** $E_p X = E_q X$ **and** $E_p 1 = E_q 1$. $\square$

Corollary: ***For*** X ***as above,***

$$H(X) \le (\mu+1)\log(\mu+1) - \mu\log\mu$$

Source Codes

- A ***source code*** C for a random variable X is a mapping $C : A \rightarrow D^*$, where D is a finite ***coding alphabet*** of size d, and D^* is the set of finite strings over D
  - **Definitions:** $C(x)$ = codeword corresponding to x, $l(x) = |C(x)|$ (length)
- The ***expected length*** of $C(x)$, for $X \sim p(x)$, is

$$L(C) = E_p[l(x)] = \Sigma_x\, p(x)\, l(x)$$

Examples:

$A=\{a,b,c,d\}$, $D = \{0,1\}$

$p(a) = 1/2$	$C(a) = 0$
$p(b) = 1/4$	$C(b) = 10$
$p(c) = 1/8$	$C(c) = 110$
$p(d) = 1/8$	$C(d) = 111$

$H(X) = 1.75$ bits $L(C) = 1.75$ bits

in fact, we have $l(x) = -\log p(x)$ for all $x \in A$ in this case

$A=\{a,b,c\}$, $D = \{0,1\}$

$p(a) = 1/3$	$C(a) = 0$
$p(b) = 1/3$	$C(b) = 10$
$p(c) = 1/3$	$C(c) = 11$

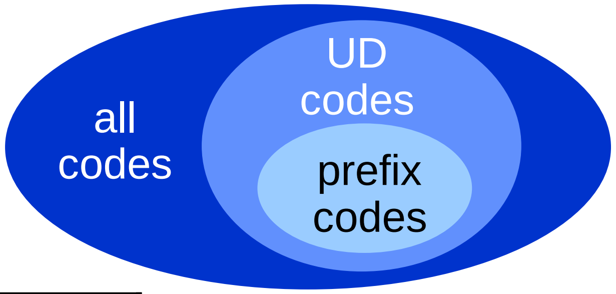
$H(X) = \log 3 \approx 1.58$ bits
$L(C) = 5/3 \approx 1.66$ bits

Source Codes (cont.)

- A code $C : A \rightarrow D^*$ **extends naturally to a code** $C^*: A^* \rightarrow D^*$ **defined by**

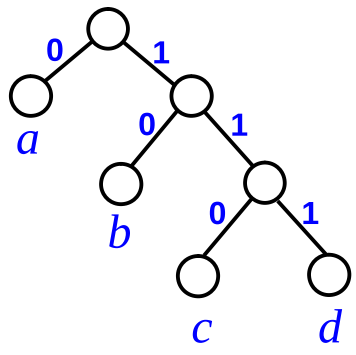
$$C^*(\lambda) = \lambda, \qquad C^*(x_1\, x_2 \ldots x_n) = C(x_1)\, C(x_2) \ldots C(x_n)$$

- C **is called *uniquely decodable* (*UD*) if its extension** C^* **is injective**
- C **is called a *prefix* (or *instantaneous*) *code* if no codeword of** C **is a prefix of any other codeword**
  - **a prefix code is uniquely decodable**
  - **prefix codes are "self-punctuating"**



Code examples

X	**not UD**	**UD, not prefix**	**prefix code**
a	0	10	0
b	010	00	10
c	01	11	110
d	10	110	111
sample string	010 $\rightarrow$ ad $\searrow$ b	100011000111... $a\ \ b\ \ d\ \ b\ \ c\ \ \ldots$	100011000111... $b\ aa\ c\ aa\ d$



a prefix code can always be described by a tree with codewords at the leaves

Prefix Codes

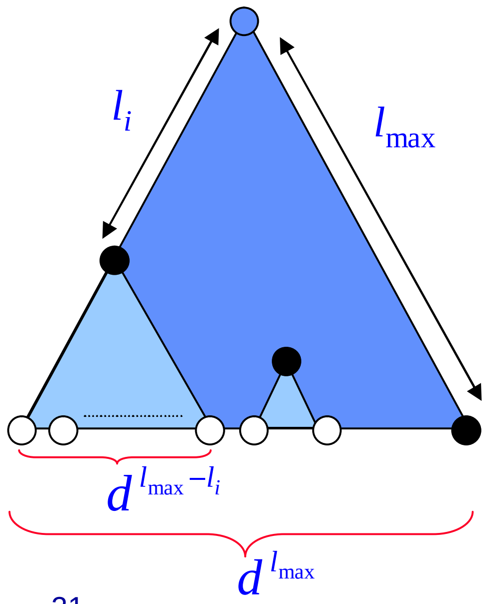
- ***Kraft's inequality***
  - **The codeword lengths $l_1, l_2, \ldots, l_m$ of any d-ary prefix code satisfy**

$$\sum_{i=1}^{m} d^{-l_i} \le 1$$

Conversely, given a set of lengths that satisfy this inequality, there exists a prefix code with these word lengths

- the theorem holds also for *countably infinite codes*
- in fact, the theorem holds for *any UD code* (McMillan)

Code tree embedded in full d-ary tree of depth $l_{\max}$



- ● inner nodes
- ● leaves
- ○ outside code

$$\sum_i d^{l_{\max}-l_i} \le d^{l_{\max}}$$

The Entropy Lower Bound

- The ***expected code length*** L **of any prefix code for a PMF** p **with probabilities** $p_1, p_2, \ldots, p_m$ **satisfies**

$$\boxed{L(C) \geq H_d(p) = \frac{H(p)}{\log d}}$$

d-ary entropy, in "dits/symbol"

with equality iff $\{ p_i \} = \{ d^{-l_i} \}$

- **Proof:** **Let** $c = \Sigma\, d^{-l_i} \leq 1$ **(Kraft),** $q_i = c^{-1} d^{-l_i}$ **(normalized distribution)**

$$L - H_d(p) = \sum_i p_i l_i + \sum_i p_i \log_d p_i =$$

$$- \sum_i p_i \log_d d^{-l_i} + \sum_i p_i \log_d p_i =$$

$$\sum_i p_i \log_d \frac{p_i}{q_i} + \log_d \frac{1}{c} = D(p \| q) + \log_d \frac{1}{c} \geq 0 \quad \text{g}$$

From now on, we assume $d = 2$ for simplicity (binary codes)

Lossless Source Coding

2. Basic coding techniques

The Shannon Code

q **The lower bound** $L(C) \geq H(p)$ **would be attained if we could have a code with lengths** $l_i = -\log p_i$ **.**
But the l_i **must be integers, and** $-\log p_i$ **are generally not**

q **Simple approximation: take** $l_i = \lceil -\log p_i \rceil$
Lengths satisfy Kraft: $\sum 2^{-l_i} \leq \sum 2^{\log p_i} = \sum p_i = 1$
Þ **there is a prefix code with these lengths (*Shannon code*)**

q **Optimal for *dyadic* distributions: all** p_i**'s powers of** 2 Þ $L = H(p)$

l **not optimal in general**

q **In general, the Shannon code satisfies**

$$L = \sum_i p_i \lceil -\log p_i \rceil \leq \sum_i p_i (-\log p_i + 1) = H(p) + 1$$

Þ **the optimal prefix code satisfies** $H(p)$ £ L £ $H(p)+1$

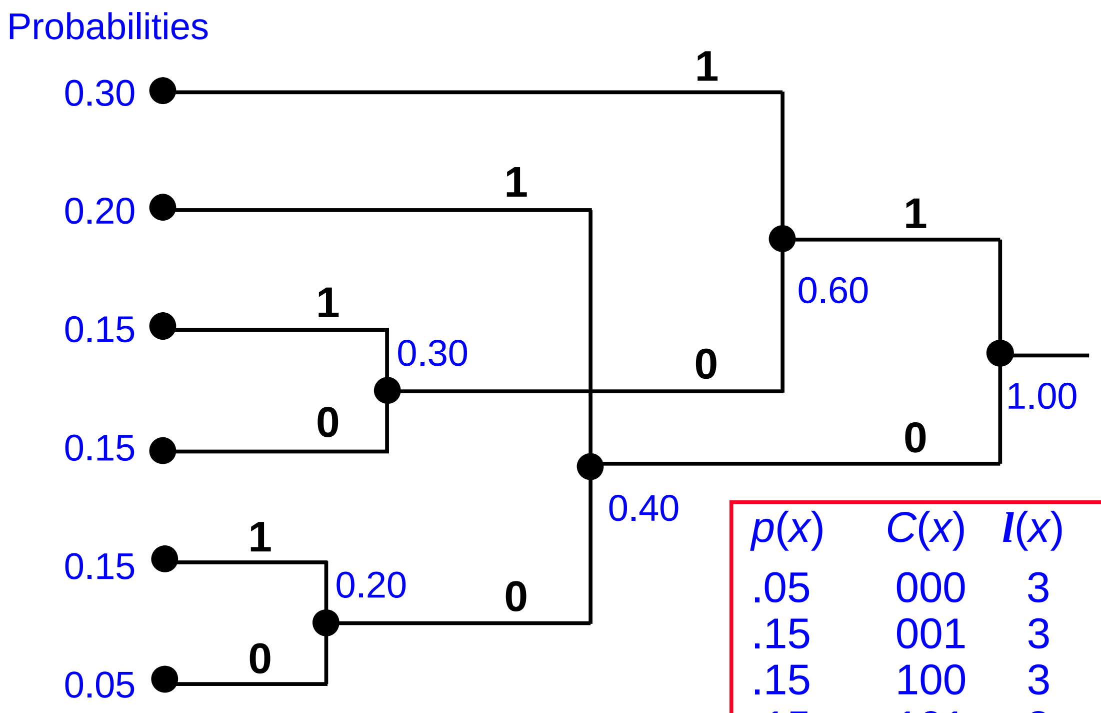
q **Upper bound cannot be improved:**

L ³ $l_{\min}$ ³ 1 **but we can have** $H(p) \rightarrow 0$

Huffman Codes

- Shannon codes are very simple but generally sub-optimal. In 1952, *Huffman* presented a construction of optimal prefix codes.

Construction of Huffman codes - by example:



$p(x)$	$C(x)$	$l(x)$
.05	000	3
.15	001	3
.15	100	3
.15	101	3
.20	01	2
.30	11	2

L=2.5 H =2.433...

Huffman algorithm

Given $p_1, p_2, \ldots, p_m$:

1. $k \leftarrow m+1$
2. find smallest pair of unused p_i, p_j
3. form $p_k = p_i + p_j$
4. mark p_i, p_j 'used'
5. if only unused is p_k **stop**
6. $k \leftarrow k+1$, **go to 2.**

Huffman Codes

Theorem: ***Codes constructed with the Huffman algorithm are optimal; i.e., if C^* is a Huffman code for a PMF p, and C is a prefix code with the same number of words, then $L_p(C^*) \le L_p(C)$.***

l Let $p_1 \ge p_2 \ge \ldots \ge p_m$ be the probabilities in p

Lemma: For any PMF, there is an optimal prefix code satisfying

1. $p_i > p_j$ Þ $l_i \le l_j$
2. the two longest codewords have the same length, they differ only in the last bit, and they correspond to the least likely symbols

Huffman codes satisfy the Lemma by construction

Proof of the Theorem: By induction on m. Trivial for $m=2$. Let C_m be a Huffman code for p. W.l.o.g., the first step in the construction of C_m merged p_m and p_{m-1}. Clearly, the remaining steps constructed a Huffman code C_{m-1} for a PMF p' with probabilities $p_1, p_2, \ldots, p_{m-2}, p_{m-1}+p_m$. Now,

$$L(C_{m-1}) = \sum_{i=1}^{m-2} l_i p_i + (l_{m-1} - 1)(p_{m-1} + p_m) = L(C_m) - p_{m-1} - p_m$$

Let C'_m be an optimal code for p, and satisfying the Lemma. Applying the same merging on C'_m, we obtain a code C'_{m-1} for p', with $L(C'_m) = L(C'_{m-1}) + p_{m-1}+p_m$. Since C_{m-1} is optimal (by ind.), we must have $L(C'_{m-1}) \ge L(C_{m-1})$ Þ $L(C'_m) \ge L(C_m)$ g

Redundancy of Huffman Codes

- ***Redundancy*: excess average code length over entropy**
  - **the redundancy of a Huffman code for a PMF p satisfies**

$$0 \leq L(C) - H(p) \leq 1$$

  - **the redundancy can get arbitrarily close to 1 when $H(p) \rightarrow 0$, but how large is it typically?**

- ***Gallager [1978]* proved**

$$L(C) - H(p) \leq P_1 + c$$

where P_1 is the probability of the most likely symbol, and

$$c = 1 - \log e + \log \log e \approx 0.086.$$

For $P_1 \geq 1/2$,

$$L(C) - H(p) \leq 2 - H_2(P_1) - P_1 \leq P_1$$

- **Precise characterization of the Huffman redundancy has been a very difficult problem**
  - **most recent results in [Szpankowsky, IEEE IT '01]**

Example

$p(x)$	$C(x)$	$l(x)$
.05	000	3
.15	001	3
.15	100	3
.15	101	3
.20	01	2
.30	11	2

L=2.5 *H* =2.433...

r = 0.067
bound = 0.386

A Coding Theorem

- For a sequence of symbols from a data source, the *per-symbol* redundancy can be reduced by using an *alphabet extension*

$$A^n = \{\ (a_1,a_2,\ldots,a_n)\ |\ a_i \in A\ \}$$

and an optimal code C^n for *super-symbols* $(X_1,X_2,\ldots,X_n) \sim p(x_1,x_2,\ldots,x_n)$.
Then, $\mathbf{H}\,(X_1,X_2,\ldots,X_n) \le L(\mathrm{C}^n) \le \mathbf{H}\,(X_1,X_2,\ldots,X_n)+1$. Dividing by n, we get:

Coding Theorem (Shannon): ***The minimum expected codeword length per symbol satisfies***

$$H(X_1,X_2,\ldots,X_n) \le L_n^* \le H(X_1,X_2,\ldots,X_n)+\frac{1}{n}.$$

Furthermore, if X^∞ ***is a random process with an entropy rate, then***

$$L_n^* \xrightarrow[n\to\infty]{} H(X^\infty)$$

Shannon tells us that there are codes that attain the fundamental compression limits asymptotically. But, how do we get there in practice?

Ideal Code Length

- A ***probability assignment*** is a function $P : A^* \rightarrow [0,1]$ satisfying

  $$\Sigma_{a\in A} P(sa) = 1 \quad \forall\, s \in A^*, \text{ with } \quad P(\lambda) = 1$$

- P is ***not*** a PMF on A^*, but it is a PMF on any ***complete subset*** of A^*
  - ***complete subset*** = leaves of a complete tree rooted at λ, e.g., A^n
- The ***ideal code length*** for a string x_1^n relative to P is defined as

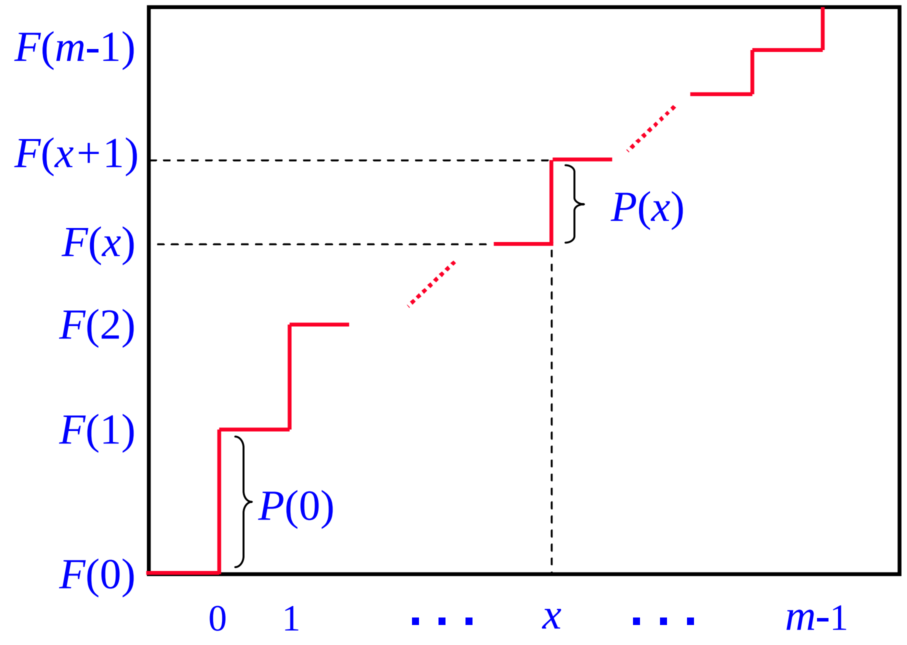
  $$l^*(x_1^n) = -\log P(x_1^n)$$

- The Shannon code attains the ideal code length for every string x_1^n, up to an integer-constraint excess $o(1)$ which we shall ignore
  - notice that attaining the ideal code length point-wise for every string is a stronger requirement than attaining the entropy on the average
- The Shannon code, as defined, is infeasible in practice (as would be a Huffman code on A^n for large n)
  - while the code length for x_1^n is relatively easy to compute given $P(x_1^n)$, it is not clear how the codeword assignment proceeds
  - as defined, it appears that one needs to look at the whole x_1^n before encoding; we would like to encode ***sequentially*** as we get the x_i

evolution that led to the solution of both issues $\Rightarrow$ ***arithmetic coding***

The Shannon-Fano Code

- **A codeword assignment for the Shannon code**
  - **Let** $X \sim P(x)$ **take values in** $M=\{0,1,\ldots,m\text{-}1\}$, $P(0) \geq P(1) \geq \ldots \geq P(m\text{-}1) > 0$
  - **Define** $F(x) = \sum_{a<x} P(a)\,,\ x \in M$ $\quad F$ ***is strictly increasing***



- **Encode** x **with the real number** $C(x) = F(x)$ **truncated to**

$$l_x = \lceil -\log P(x) \rceil \text{ bits}$$

(digits to the right of the binary point)

  - C **is prefix-free**
  - $C(x)$ **is in the interval**

$$F(x\text{-}1) < C(x) \leq F(x)$$

Example:

x	P	F	l_x	$C(x)$
0	0.5	0	1	.0
1	0.25	0.5	2	.10
2	0.125	0.75	3	.110
3	0.125	0.875	3	.111

Elias Coding - Arithmetic Coding

- **To encode x_1^n we take $M = A^n$, ordered lexicographically**
  - **to compute $F(x_1^n)$ directly, we would need to add an exponential number of probabilities, and compute with huge precision -- infeasible**
- ***Sequential probability assignment***

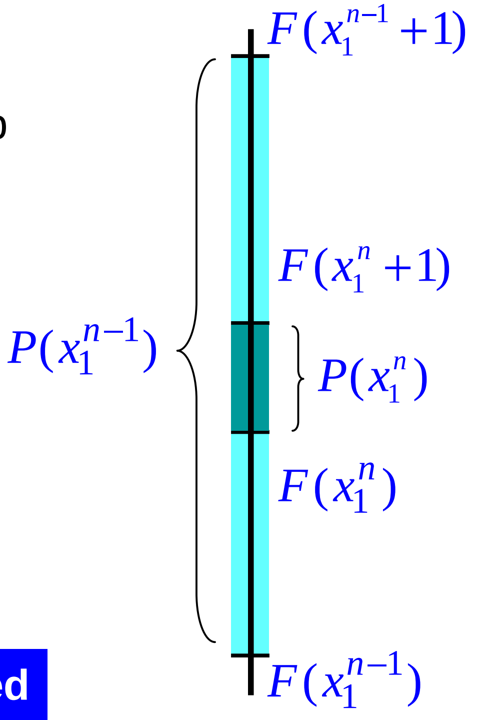
$$P(x_1^n) = P(x_1^{n-1})\underbrace{P(x_n \mid x_1^{n-1})}$$

what the model will provide at each step

- ***Sequential encoding***

$$F(x_1^n) = \sum_{y_1^n < x_1^n} P(y_1^n) = \sum_{y_1^{n-1} < x_1^{n-1}} P(y_1^{n-1}) + \sum_{y < x_n} P(x_1^{n-1} y)$$

$$\Rightarrow \boxed{F(x_1^n) = F(x_1^{n-1}) + P(x_1^{n-1}) \sum_{y < x_n} P(y \mid x_1^{n-1})}$$



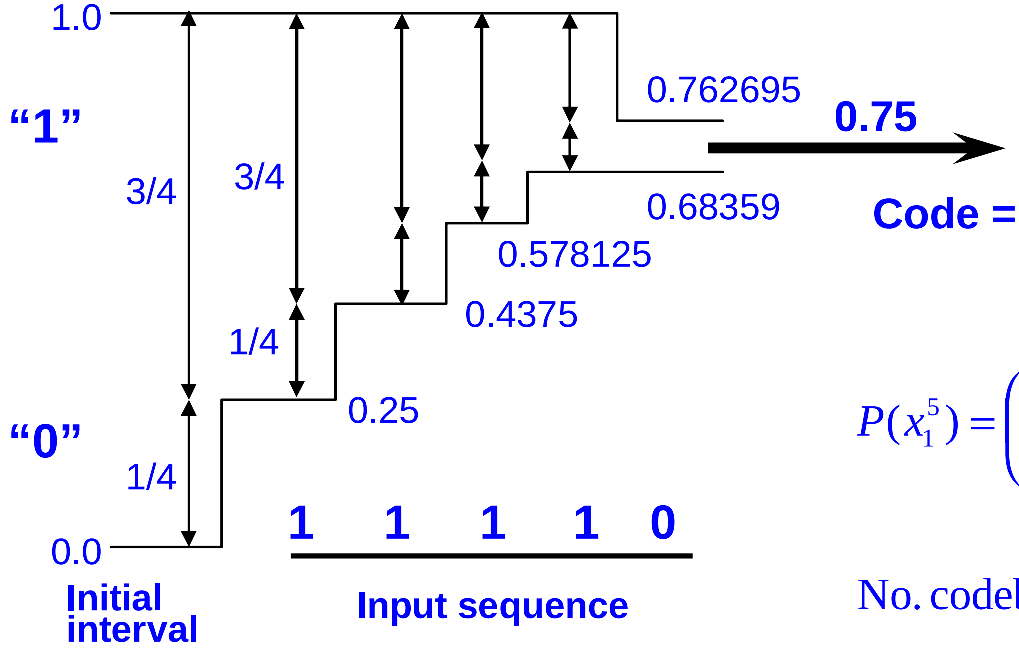
The "active" interval shrinks, it has width $P(x_1^n)$,
and, as $n \to \infty$, it converges to the real number $F(x_1^\infty)$

x_1^∞ is encoded by means of one real number, computed sequentially by arithmetic operations

$\Rightarrow$ *arithmetic coding*

Arithmetic Coding - Example

***P*(0) = 0.25   *P*(1) = 0.75 (static i.i.d. model)**



$$P(x_1^5) = \left(\frac{3}{4}\right)^4 \cdot \frac{1}{4} = \frac{81}{1024} = 0.0791$$

$$\text{No. codebits} = \left\lceil -\log_2 P(x_1^5) \right\rceil = 4$$

- **Computational challenges**
  - **precision of floating-point operations – *register length***
  - ***active interval shrinks, but small numerical changes can lead to changes in many bits of the binary representation – carry-over problem***
  - ***encoding/decoding delay* – how many cycles does it take since a digit enters the encoder until it can be output by the decoder?**

Arithmetic Coding

- ***Arithmetic coding* [Elias ca.'60, Rissanen '75, Pasco '76] solves problems of precision and carry-over in the sequential computation of $F(x_1^n)$, making it practical with bounded delay and modest memory requirements**
  - **refinements and contributions by many researchers in past 25 years**
- **When carefully designed, AC attains a code length**

$$-\log P(x_1^n) + O(1),$$

  ideal up to an additive constant
- **It reduces the lossless compression problem to one of finding the best probability assignment for the given data x_1^n, that which will provide the shortest ideal code length**

the problem is not to find the best code for a given probability distribution, it is to find the best probability assignment for the data at hand

Lossless Source Coding

4. Lempel-Ziv coding

The Lempel-Ziv Algorithms

- **A family of data compression algorithms first presented in**

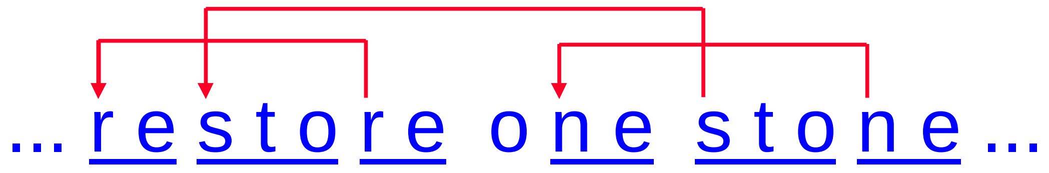
  [LZ77] J. Ziv and A. Lempel, “A universal algorithm for sequential data compression,” *IEEE Trans. Inform.Theory*, vol. IT-23, pp. 337–343, May 1977

  [LZ78] J. Ziv and A. Lempel, “Compression of individual sequences via variable rate coding,” *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 530–536, Sept. 1978.

- **Many desirable features, the conjunction of which was unprecedented**
  - ***simple* and *elegant***
  - ***universal* for *individual sequences* in the class of *finite-state encoders***
    - Arguably, every real-life computer is a finite-state automaton
  - **convergence to the entropy for *stationary ergodic sources***
  - ***string matching* and *dictionaries*, no explicit probability model**
  - **very *practical*, with *fast and effective implementations* applicable to a wide range of data types**

Two Main Variants

- [LZ77] and [LZ78] present different algorithms with common elements
  - The main mechanism in both schemes is *pattern matching*: find string patterns that have occurred in the past, and compress them by encoding a reference to the previous occurrence



- Both schemes are in wide practical use
  - many variations exist on each of the major schemes
  - we focus on LZ78, which admits a simpler analysis with a stronger result. The proof here follows [Cover & Thomas '91] , attributed to [Wyner & Ziv]. It differs from the original proof in [LZ78].
  - the scheme is based on the notion of *incremental parsing*

Incremental Parsing

q **Parse the input sequence x_1^n into *phrases*, each new phrase being the shortest substring that has not appeared so far in the parsing**

x_1^n **= 1,0,1 1,0 1,0 1 0,0 0,1 0, ...** **(assume $A=\{0,1\}$)**

1 2 3 4 5 6 7

q **Each new phrase is of the form $\mathbf{w}b$, $\mathbf{w}$ = a previous phrase, $b\in\{0,1\}$**

l **a new phrase can be described as (i,b), where $i = index\,(w)$ (phrase #)**

l **in the example: (0,1), (0,0), (1,1), (2,1), (4,0), (2,0),(1,0) (phrase #0 = λ)**

l **let $c(n)$ = number of phrases in x_1^n**

l **a phrase description takes $\leq 1+\log c(n)$ bits**

l **in the example, 28 bits to describe 13 : bad deal! it gets better as $n\rightarrow\infty$**

l **decoding is straightforward**

l **in practice, we do not need to know $c(n)$ before we start encoding**

u use increasing length codes that the decoder can keep track of

Lemma:

$$c(n) \leq \frac{n}{(1-e_n)\log n}, \quad e_n \rightarrow 0 \text{ as } n\rightarrow\infty$$

Proof: $c(n)$ is max when we take all phrases as short as possible. Let $n_k = \sum_{j=1}^{k} j2^j = (k-1)2^{k+1}+2,$

with $n_k \leq n < n_{k+1}$. Then $c(n) \leq n/(k\text{-}1) = n/[(1-e)\log n]$ with $e = O(\log\log n/\log n)$. n

Universality of LZ78

- Let $Q_k(x_{-(k-1)},\ldots,x_{-1},x_0,x_1,\ldots,x_n) \overset{\Delta}{=} Q(x_{-(k-1)}^0)\prod_{j=1}^{n} Q(x_j \mid x_{j-k}^{j-1})$

  be *any* k-th order Markov probability assignment for x_1^n, with arbitrary initial state $(x_{-(k-1)},\ldots,x_0)$
- Assume x_1^n is parsed into distinct phrases $y_1, y_2,\ldots,y_c$. Define:

where are we going with all this?

- v_i = index of start of $y_i = (x_{v_i},\ldots,x_{v_{i+1}-1})$
- $s_i = (x_{v_i-k},\ldots,x_{v_i-1})$ = the k bits preceding y_i in x_1^n, $s_1 = (x_{-(k-1)},\ldots,x_0)$
- c_{ls} = number of phrases y_i of length l and preceding state $s \in \{0,1\}^k$
- we have $\Sigma_{l,s}\, c_{ls} = c$ and $\Sigma_{l,s}\, l\, c_{ls} = n$

Ziv's inequality: For any distinct parsing of x_1^n, and any Q_k, we have

$$\log Q_k(x_1,\ldots,x_n \mid s_1) \le -\sum_{l,s} c_{ls}\log c_{ls}$$

The lemma upperbounds the probability of *any sequence* under *any probability assignment* from the class, based on properties of *any distinct parsing* of the sequence (including the incremental parsing)

Universality of LZ78 (proof of Ziv's inequality)

Proof of the Ziv's inequality:

$$Q_k(x_1, x_2, \ldots, x_n \mid s_1) = Q(y_1, y_2, \ldots, y_c \mid s_1) = \prod_{i=1}^{c} Q(y_i \mid s_i)$$

$$\begin{aligned}
\log Q_k(x_1, x_2, \ldots, x_n \mid s_1) &= \sum_{i=1}^{c} \log Q(y_i \mid s_i) \\
&= \sum_{l,s} \sum_{i:|y_i|=l, s_i=s} \log Q(y_i \mid s_i) \\
&= \sum_{l,s} c_{ls} \sum_{i:|y_i|=l, s_i=s} \frac{1}{c_{ls}} \log Q(y_i \mid s_i) \\
\text{Jensen} \longrightarrow\ &\le \sum_{l,s} c_{ls} \log\left(\sum_{i:|y_i|=l, s_i=s} \frac{1}{c_{ls}} Q(y_i \mid s_i) \right)
\end{aligned}$$

Since the y_i **are distinct, we have** $\displaystyle\sum_{i:|y_i|=l, s_i=s} Q(y_i \mid s_i) \le 1$

$$\Rightarrow \quad \log Q_k(x_1, x_2, \ldots, x_n \mid s_1) \le \sum_{l,s} c_{ls} \log \frac{1}{c_{ls}} \qquad \text{n}$$

Universality for Individual Sequences: Theorem

Theorem: ***For any sequence*** x_1^n ***and for any*** k***-th order probability assignment*** Q_k ***, we have***

$$\frac{c(n)\log c(n)}{n} \le -\frac{1}{n}\log Q_k(x_1^n \mid s_1) + \frac{(1+o(1))k}{\log n} + O\left(\frac{\log\log n}{\log n}\right)$$

Proof: Lemma $\Rightarrow$ $\log Q(x_1^n \mid s_1) \le -\sum_{l,s} c_{ls}\log\frac{c_{ls}c}{c} = -c\log c - c\sum_{l,s}\mathrm{p}_{ls}\log \mathrm{p}_{ls}, \quad \mathrm{p}_{ls} \overset{\Delta}{=} \frac{c_{ls}}{c}$

We have $\Sigma_{l,s}\,\mathrm{p}_{ls} = 1$ and $\Sigma_{l,s}\, l\, \mathrm{p}_{ls} = n/c$. Define r.v.'s $U,S \sim P(U=l,S=s) = \mathrm{p}_{ls}$

Then, $EU = n/c$ and $-\frac{1}{n}\log Q(x_1^n \mid s_1) \ge \frac{c}{n}\log c - \frac{c}{n}H(U,V) \ge \frac{c}{n}\log c - \frac{c}{n}(H(U)+H(V))$

Now, $H(V) \le k$, and by the maximum entropy theorem for mean-constrained r.v.'s,

$$H(U) \le \left(\frac{n}{c}+1\right)\log\left(\frac{n}{c}+1\right) - \frac{n}{c}\log\frac{n}{c} \quad \Rightarrow \quad \frac{c}{n}H(U,V) \le \frac{c}{n}k + \frac{c}{n}\log\frac{n}{c} + o(1)$$

Recall $c/n \le (1+\mathrm{o}(1))/\log n$ $\Rightarrow$ $\frac{c}{n}\log\frac{n}{c} \le O\left(\frac{\log\log n}{\log n}\right)$

$$\Rightarrow \quad -\frac{1}{n}\log Q\left(x_1^n \mid s_1\right) \ge \frac{c\log c}{n} - \frac{(1+o(1))k}{n} - O\left(\frac{\log\log n}{\log n}\right)$$ n

Universality for Individual Sequences: Discussion

- **The theorem holds for *any k-th order probability assignment* Q_k, and in particular, for the k-th order empirical distribution of x_1^n, which gives an ideal code length equal to the empirical entropy**

$$-\frac{1}{n}\log \hat{P}(x_1^n) = \hat{H}(x_1^n)$$

- **The asymptotic $O(\log\log n/\log n)$ term in the redundancy has been improved to $O(1/\log n)$ – no better upper bound can be achieved**
  - **obtained with tools from *renewal theory***

Compressibility

Q_k **is optimized for** x_1^n**,** ***for each*** k

- ***Finite-memory compressibility***

we must have n®¥ before k®¥, otherwise definitions are meaningless!

$$FM_k(x_1^n) = \inf_{Q_k, s_1}\left(-\frac{1}{n}\log Q_k\left(x_1^n \mid s_1\right)\right)$$ k**-th order, finite sequence**

$$FM_k(x_1^\infty) = \limsup_{n\to\infty}\left(FM_k(x_1^n)\right)$$ k**-th order, infinite sequence**

$$FM(x_1^\infty) = \lim_{k\to\infty} FM_k(x_1^\infty)$$ ***FM compressibility***

- ***Lempel-Ziv compression ratio***

$$LZ(x_1^n) = \frac{c(n)\left(\log c(n)+1\right)}{n}$$ **finite sequence**

$$LZ(x_1^\infty) = \limsup_{n\to\infty}\left(LZ(x_1^n)\right)$$ ***LZ compression ratio***

Theorem: ***For any sequence*** x_1^∞, $LZ(x_1^\infty) \le FM(x_1^\infty)$

Probabilistic Setting

Theorem: ***Let*** $X_{-\infty}^{\infty}$ ***be a stationary ergodic random process. Then,***

$$LZ(X_1^\infty) \le H(X_1^\infty) \quad \textit{with probability } 1$$

Proof: via approximation of the stationary ergodic process with Markov processes of increasing order, and the previous theorems

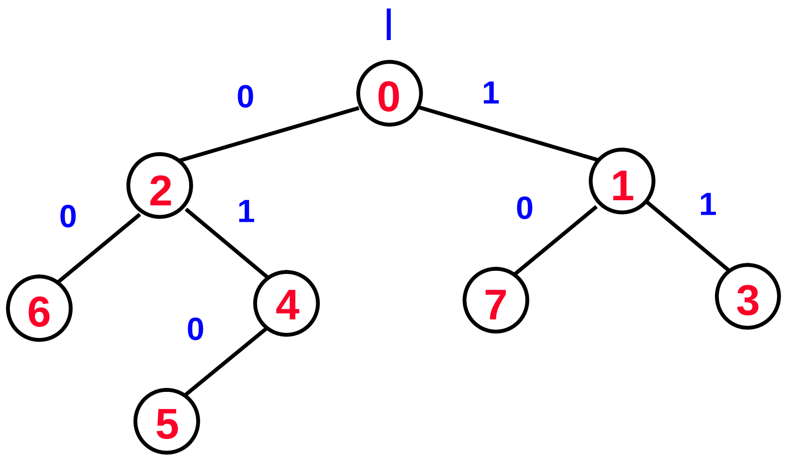
$$Q_k(x_{-(k-1)}^0 x_1^n) \overset{\Delta}{=} P_X(x_{-(k-1)}^0) \prod_{j=1}^{n} P_X(x_j \mid x_{j-k}^{j-1}), \quad X \sim P_X$$

Markov k**-th order approximation**

$$H(x_j \mid x_{j-k}^{j-1}) \xrightarrow{k\to\infty} H(X)$$

The Parsing Tree

x_1^n = 1,0,1 1,0 1,0 1 0,0 0,1 0, ...



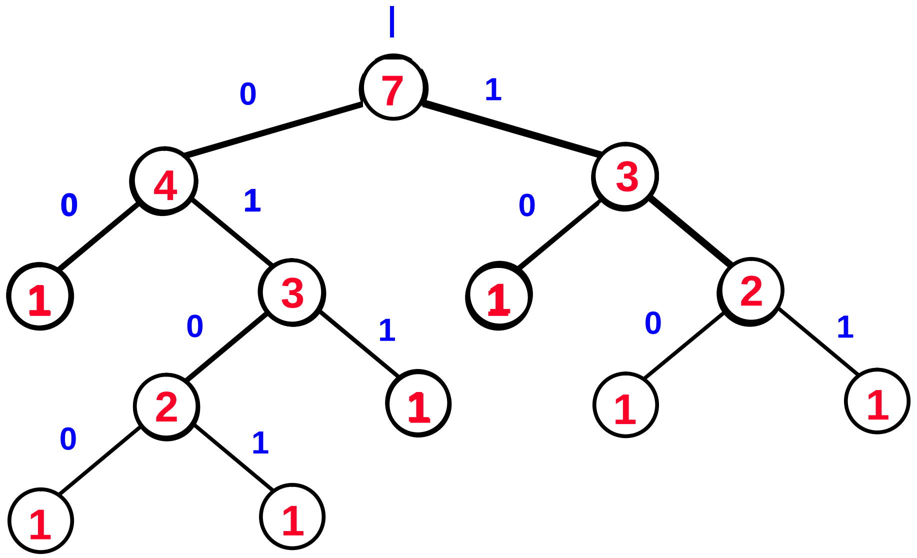
code	*phrase*
0	l
1	0,1
2	0,0
3	1,1
4	2,1
5	4,0
6	2,0
7	1,0
⋮	⋮

dictionary

- coding could be made more efficient by "recycling" codes of nodes that have a complete set of children (e.g., 1, 2 above)
- will not affect asymptotics
- many (many many) tricks and hacks exist in practical implementations

The LZ Probability Assignment

x_1^n **= 1,0,1 1,0 1,0 1 0, ...**



- **In general,**

$$P(x_1^n) = \frac{1}{(c(n)+1)!}$$

$$-\log P = c(n)\log c(n) + o\big(c(n)\log c(n)\big)$$ **LZ code length!**

- **Slightly different tree evolution** ***anticipatory parsing***
- **A *weight* is kept at every node**
  - **number of times the node was traversed through + 1**
- **A node act as a conditioning state, assigning to its children probabilities proportional to their weight**
- **Example: string** s=101101010
  - $P(0|s)$ = 4/7
  - $P(1|s0)$ = 3/4
  - $P(1|s01)$ = 1/3
  - $P(011|s)$ = (4/7)*(3/4)*(1/3) = 1/7

  Notice `telescoping'
- $P(s011)$ = 1/7!

every lossless compression algorithm defines a prob. assignment, even if it wasn't meant to!

Other Properties

q **Individual sequences result applies also to FSM probability assignments**

q **The “worst sequence”**

  l ***counting sequence*** **0 1 00 01 10 11 000 001 010 011 100 101 110 111 ..**

  l **maximizes** $c(n)$ Þ **incompressible with LZ78**

q **Generalization to larger alphabets is straightforward**

q ***LZW modification*****: extension symbol** b **not sent. It is determined by the first symbol of the next phrase instead [Welch 1984]**

  l **dictionary is initialized with all single-symbol strings**

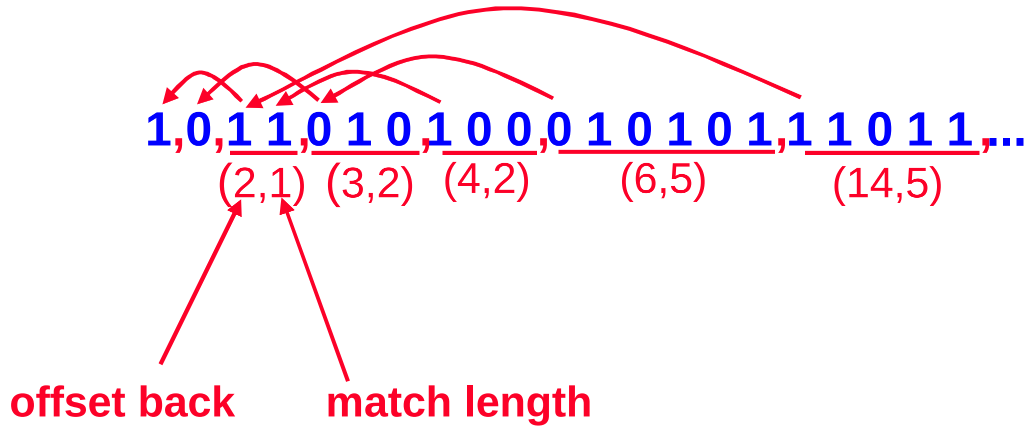
  l **works very well in practice**

  l **breakthrough in popularization of LZ, led to UNIX** ***compress***

q **In real life we use** ***bounded dictionaries*****, and need to reset them from time to time**

Lempel-Ziv 77

- ***Exhaustive parsing*** **as opposed to *incremental***
  - **a new phrase is formed by the longest match *anywhere in a finite past window,* plus the new symbol**
  - **a pointer to the location of the match, its length, and the new symbol are sent**
- **Has a weaker proof of universality, but actually works better in practice**



Lempel-Ziv in the Real World

- **The most popular data compression algorithm in use**
  - **virtually every computer in the world runs some variant of LZ**
  - **LZ78**
    - compress
    - GIF
    - TIFF
    - V.42 modems
  - **LZ77**
    - gzip, pkzip (LZ77 + Huffman for pointers and symbols)
    - png
  - **many more implementations in software and hardware**
    - MS Windows dll - software distribution
    - tape drives
    - printers
    - network routers
    - various comemrcially available VLSI designs
    - ...