Applications of Information Theory in Image Processing

1. Review of Information Theory and Lossless Source Coding (part B)

Notation

- A : discrete (usually finite) alphabet; $\alpha = |A|$: size of A (when finite)
- A^n : set of strings of length n over A; A^* : set of finite strings over A
- λ : empty string
- $x_1^n = x^n = x_1 x_2 \ldots x_n$: finite sequence over A
- $x_1^\infty = x^\infty = x_1 x_2 \ldots x_t \ldots$: infinite sequence over A
- $x_i^j = x_i x_{i+1} \ldots x_j$: sub-sequence (i sometimes omitted if $= 1$)
- $p_X(x)$ or $P_X(x)$: $\text{Prob}(X = x)$ probability mass function (PMF) on A
  (subscript 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)
- $\hat{p}_{x^n}(x)$: empirical distribution obtained from x^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 r.v. X or PMF p, in bits (usually per-symbol)
- $\mathbf{H}(X^n)$: joint entropy of $X_1, X_2, \ldots, X_n$ (unnormalized)
- $H_2(p) = -p \log p - (1-p) \log(1-p)$, $0 \le p \le 1$: binary entropy function
- $D(p||q)$: relative entropy (information divergence) between PMFs p and q

Source Codes

- A *source code* C for a random variable X defined on an alphabet A is a mapping $C : A \to D^*$, where D is a finite *coding alphabet* of size d, and D^* is the set of finite strings over D (often, $d = 2,\ D = \{0, 1\}$).
  - $C(x) =$ *codeword* corresponding to $x \in A$ (finite string over D)
  - $\ell(x) = |C(x)|$ *code length*
- Let $X \sim p(x)$. The *expected length* of $C(X)$ is

$$L(C) = E_p[\ell(X)] = \sum_x p(x)\ell(x)$$

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 $\quad L(C) = 1.75$ bits

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

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

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

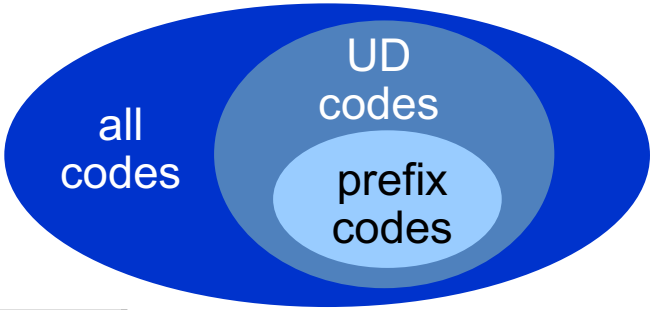
$H(X) = \log 3 \approx 1.58$ bits
$L(C) = 5/3 \approx 1.66$ bits

Source Codes (cont.)

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

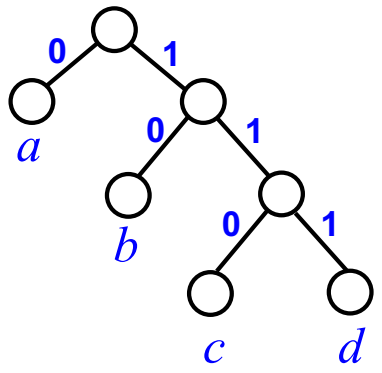
$$C^*(\lambda) = \lambda, \quad C^*(x_1 x_2 \dots x_n) = C(x_1)C(x_2) \dots C(x_n).$$

- C is called *uniquely decodable* (*UD*) if its extension C^* is injective (1-1).
- 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 → ad or b	10 00 110 00 1110 ... a b d b c a ...	10 00 110 00 1110 ... b aa c aa d a ...



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

Prefix Codes

❑ *Kraft's inequality*

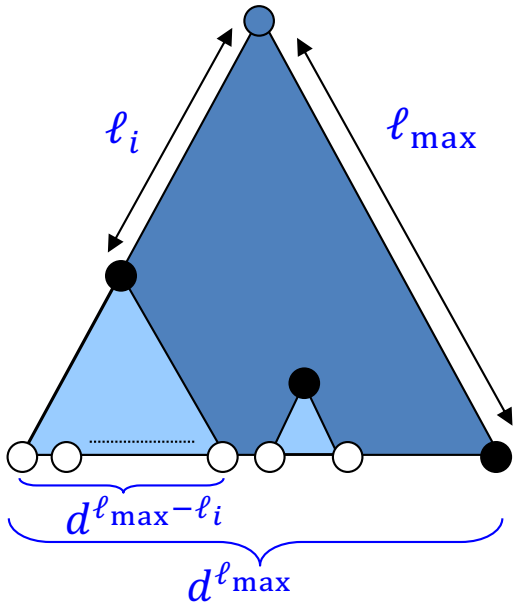
- The codeword lengths $\ell_1, \ell_2, \ldots, \ell_m$ of any d-ary prefix code satisfy

$$\sum_{i=1}^{m} d^{-\ell_i} \leq 1.$$

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

- in fact, the theorem holds for *any UD code* (McMillan)
- ⇒ *no advantage in using UD codes that are not prefix codes*
- codes with Kraft sum < 1 are always *sub-optimal*

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



- inner nodes
- leaves
- outside code

$$\sum_i d^{\ell_{\max}-\ell_i} \leq d^{\ell_{\max}}$$

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

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

d-ary entropy, in "dits/symbol"

with equality iff $\ell_i = -\log_d p_i$, $1 \leq i \leq m$.

- **Proof**: Let $c = \sum_i d^{-\ell_i}$ (Kraft), $q_i = c^{-1} d^{-\ell_i}$ (normalized distribution).

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

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

$\log_d d^{-\ell_i} = \log_d q_i + \log_d c$

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

■

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

The Shannon Code

- The lower bound $L(C) \geq H(p)$ would be attained if we could have a code with lengths $\ell_i = -\log p_i$.

  But the ℓ_i must be integers, and $-\log p_i$ are generally not.

- Simple approximation: take $\ell_i = \lceil -\log p_i \rceil$.

- Lengths satisfy Kraft:

$$\sum_i 2^{-\ell_i} = \sum_i 2^{-\lceil -\log p_i \rceil} \leq \sum_i 2^{\log p_i} = \sum_i p_i = 1.$$

  $\Longrightarrow$ there is a prefix code with these lengths (*Shannon code*).

- Optimal for *dyadic* distributions:

  all p_i's powers of $2 \Longrightarrow L = H(p)$

  - not optimal in general

The Shannon Code (cont.)

- In general, the Shannon code satisfies

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

$\Longrightarrow$ the optimal prefix code satisfies $H(p) \leq L \leq H(p) + 1$.

- Upper bound cannot be improved:

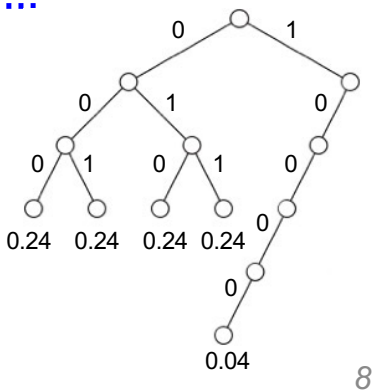
$L \geq \ell_{\min} \geq 1$ but we can have $H(p)$ arbitrarily close to 0.

Example

p_i	$\lceil -\log p_i \rceil$	code
0.24	3	000
0.24	3	001
0.24	3	010
0.24	3	011
0.04	5	10000

Kraft: $\sum 2^{-\ell_i} = 4 \cdot 2^{-3} + 2^{-5} = \frac{17}{32} < 1$ (far!)

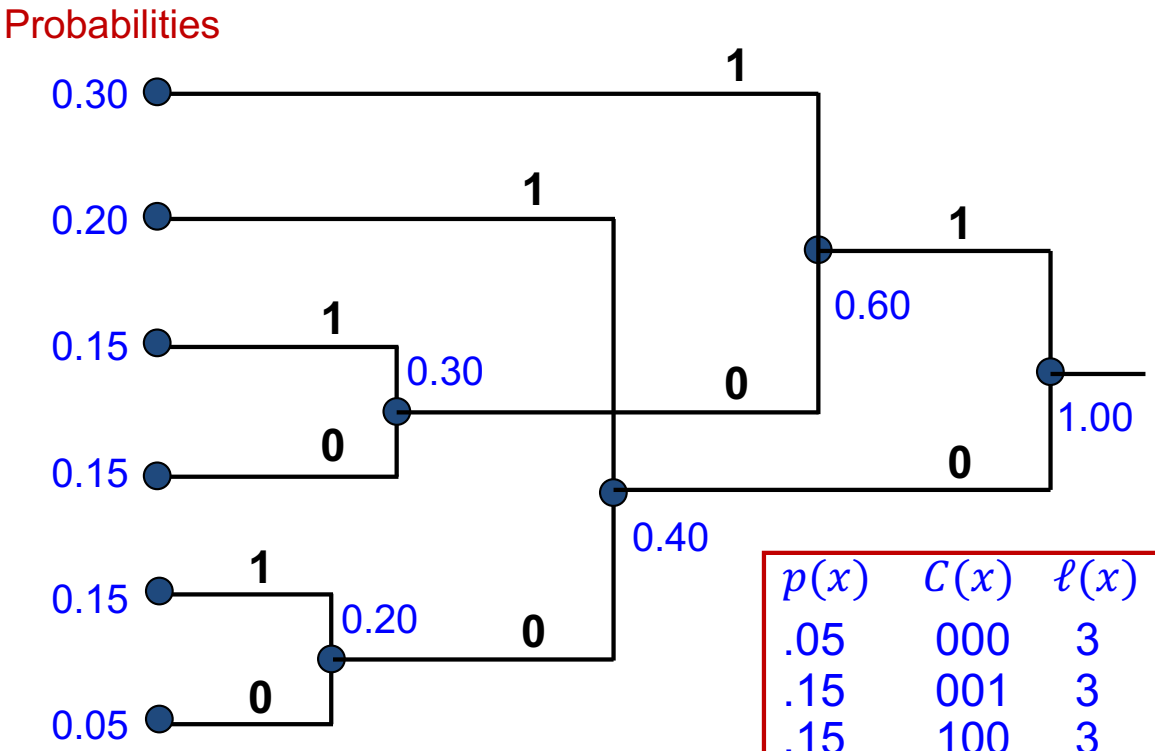
$L(C) = 3.08 \quad H = 2.162 \ldots$



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:



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

$L = 2.5 \quad H = 2.433\ldots$

Shannon: 2.8

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 $L_p(C^*) \leq L_p(C)$.*

Let $p_1 \geq p_2 \geq \cdots \geq p_m$ be the probabilities in p .

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

1. $p_i > p_j \Rightarrow \ell_i \leq \ell_j$
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,

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

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

$$L(C'_{m-1}) \geq L(C_{m-1}) \Rightarrow L(C'_m) \geq L(C_m).$$

■

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 $H(p) \to 0$, 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. There has been progress, but the general problem *is still open.*

Example

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

$L = 2.5$
$H = 2.433 \ldots$

r = 0.067
bound = 0.386

Entropy and Codes: Summary

- Entropy: $X \sim p(x)$, $H(X) = -\sum_{x \in A} p(x) \log p(x) \leq \log |A|$ (also $H(p)$).
- Joint entropy: $\mathbf{H}(X^n) = -\sum_{x^n \in A^n} p(x^n) \log p(x^n) \leq n \log |A|$.
- Entropy rate: $\lim_{n \to \infty} \frac{1}{n} \mathbf{H}(X_1, X_2, \ldots, X_n) \leq \log |A|$.
- Code C over A: $L_p(C) = \sum_{x \in A} p(x) \ell_C(x)$
  - For UD codes, no performance loss in restriction to prefix codes.
- Entropy lower bound:

$$L_p(C) \geq H(p)$$

  - Equality iff $p(x) = 2^{-\ell} \quad \forall x \in A$.
- Shannon code: $\ell(x) = \lceil -\log p(x) \rceil$
  - sub-optimal, Kraft sum generally < 1.
- Huffman codes: optimal prefix codes, best possible $L_p(C)$.

$$H(p) \leq L_p(C_{\text{Huffman}}) \leq L_p(C_{\text{Shannon}}) \leq H(p) + 1$$

- Problem with symbol by symbol encoding: can never get below compression ratio of 1 bit/symbol. Would like to approach entropy rate.

A Coding Theorem

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

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

and an optimal code C^n for *super-symbols*

$$(X_1, X_2, \dots, X_n) \sim p(x_1, x_2, \dots, x_n).$$

Then, $\mathbf{H}(X_1, X_2, \dots, X_n) \leq L(C^n) \leq \mathbf{H}(X_1, X_2, \dots, X_n) + 1.$

Dividing by n, we get:

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

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

Furthermore, if X^∞ *is a random proces 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?

Probability Assignments and Ideal Code Length

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

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

- 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
- Implicit *sequentiality*

$$1 = \frac{\sum_a P(sa)}{P(s)} = \sum_a \frac{P(sa)}{P(s)} \stackrel{\text{def}}{=} \sum_a P(a|s)$$

- The *ideal code length* for a string x_1^n relative to P is defined as

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

Ideal Code Length (cont.)

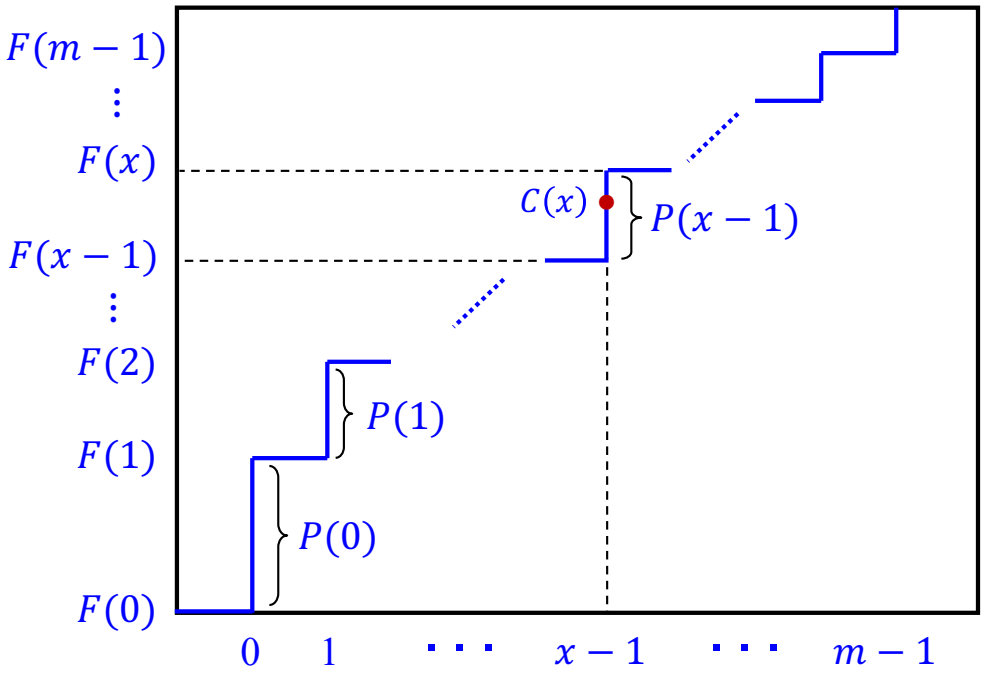
$$\ell_{\text{Shannon}}(x^n) = \lceil -\log p(x_1, x_2, \ldots, x_n) \rceil \leq -\log p(x_1, x_2, \ldots, x_n) + 1$$

- The Shannon code attains the ideal code length for *every string* x^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^n is relatively easy to compute given $P(x^n)$, it is not clear how the codeword assignment proceeds
  - as defined, it appears that one needs to look at the whole x^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 $M = \{0,1,\ldots,m-1\}$, $P(0) \geq P(1) \geq \cdots \geq P(m-1) > 0$
  - Define $F(x) = \sum_{a<x} P(a)$, $x \in M$. F is *strictly increasing*.



- Encode x with the real number

$$C(x) = \Big(F(x) \textit{ truncated to } \ell_x = \lceil -\log P(x) \rceil \text{ bits}\Big)$$

(digits to the right of the binary point)

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

$$F(x-1) < C(x) \leq F(x)$$

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

The Shannon-Fano Code (cont.)

- To encode x^n, we take $M = A^n$, ordered lexicographically.
- Challenges in the computation of $F(x^n) = \sum_{y^n < x^n} P(y^n)$
  - it seems that we need to add an exponential number of probabilities, with huge precision
  - it still seems that we need to have the full sequence x^n before we can start: *no sequentiallity*
- Let $A = \{0, 1, 2, \ldots, \alpha - 1\}$, $x_n, x_{n-1} > 0$. Sequences whose $P(\cdot)$ we need to add:

1	2	3	$\cdots$	$\cdots$	$n-1$	n
0	0	0	$\cdots$	$\cdots$	0	0
1	0	0	$\cdots$	$\cdots$	0	0
2	0	0	$\cdots$	$\cdots$	0	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\alpha-1$	0	0	$\cdots$	$\cdots$	0	0
0	1	0	$\cdots$	$\cdots$	0	0
1	1	0	$\cdots$	$\cdots$	0	0
2	1	0	$\cdots$	$\cdots$	0	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
x_1	x_2	x_3	$\cdots$	$\cdots$	$x_{n-1}-1$	$\alpha-1$
x_1	x_2	x_3	$\cdots$	$\cdots$	x_{n-1}	0
x_1	x_2	x_3	$\cdots$	$\cdots$	x_{n-1}	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
x_1	x_2	x_3	$\cdots$	$\cdots$	x_{n-1}	x_n-1

all sequences $y^n < x^n$

all sequences y^n with $y^{n-1} < x^{n-1}$

sum of $P(\cdot)$ is $F(x^{n-1})$

all sequences y^n with $y^{n-1} = x^{n-1}$, $y_n < x_n$

Elias Coding - Arithmetic Coding

- *Sequential probability assignment*

$$P(x^n) = P(x^{n-1}) \cdot P(x_n|x^{n-1})$$

what we'll maintain and update

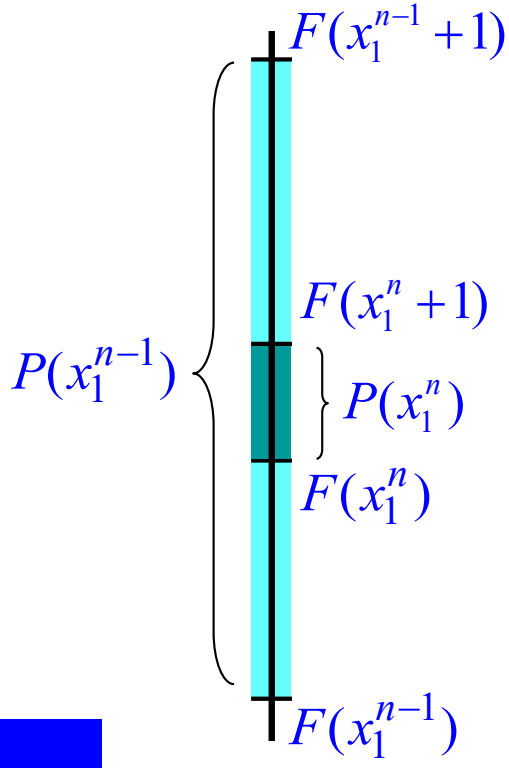
what the model will provide at each step

- *Sequential computation of $F(x_1^n)$ (encoding)*

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

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

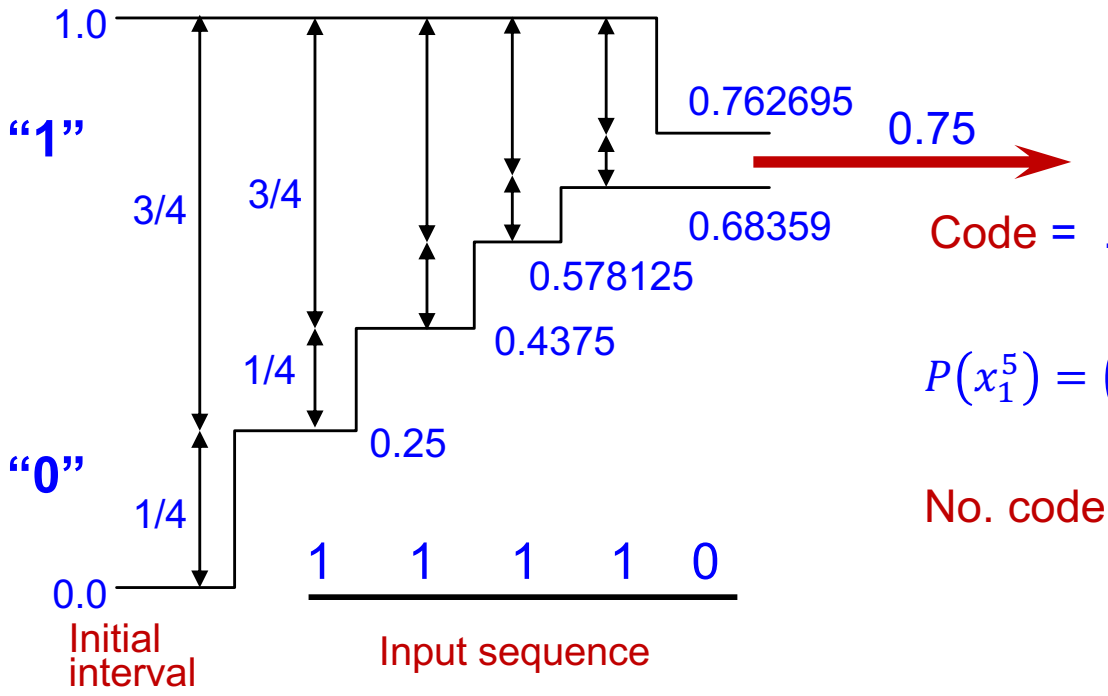
We maintain an "active" interval that shrinks, has width $P(x^n)$, and, as $n \to \infty$, it converges to the real number $F(x^\infty)$



x^∞ 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)



Code = .1100

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

No. codebits = $\lceil -\log P(x_1^5) \rceil = 4$

- 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 $F(x^n)$, making it practical with bounded delay and modest memory requirements.
  - Refinements and contributions by many researchers over the decades.
- When carefully designed, AC attains a code length

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

  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^n, that which will provide the shortest ideal code length.
- Terminology: *probability assignment = model*

Codes and probability assignments

- We saw that a probability assignment induces, through AC, a code that attains the ideal code length up to $O(1)$.
- Conversely, a code with length function $L(x^n)$ satisfying Kraft's inequality

$$\sum_{x^n} 2^{-L(x^n)} \leq 1$$

naturally defines a probability assignment

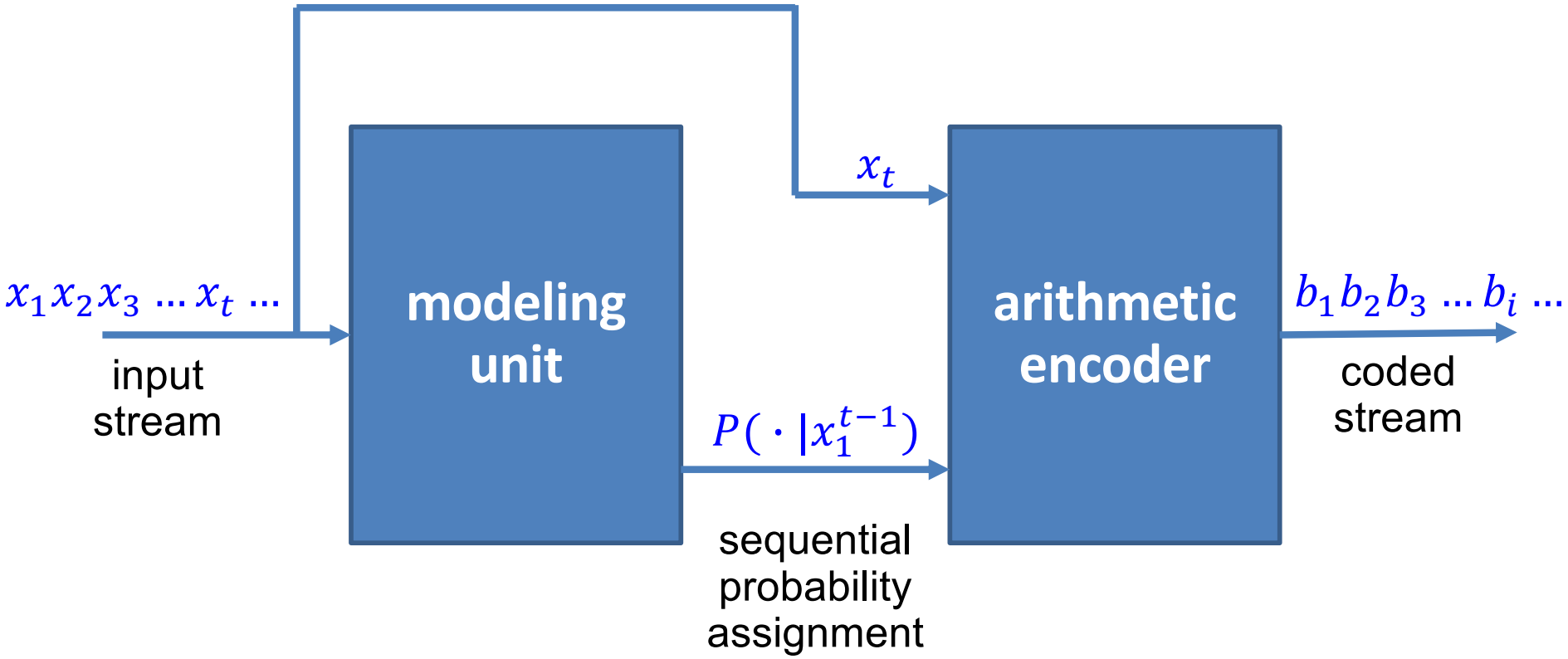
$$P_L(x^n) = 2^{-L(x^n)}$$

Codes $\Leftrightarrow$ *Probability assignments*

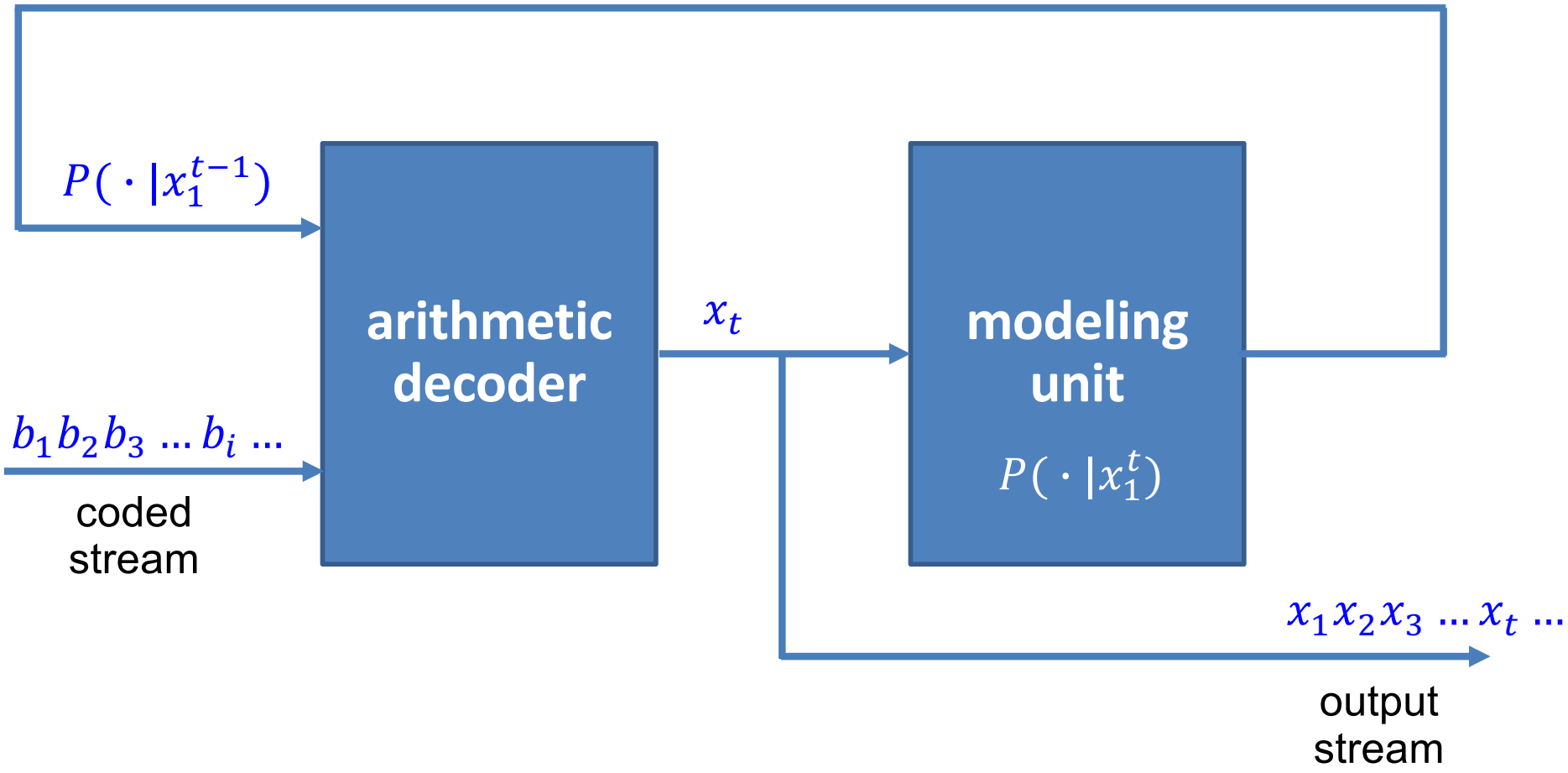
the lossless compression problem is not to find the best code for a given probability distribution, it is to find the best probability assignment for the data at hand

- *Coding system = modeling unit + coding unit*

Lossless coding system (encoder)



Lossless coding system (decoder)



Model Classes and Universal Coding

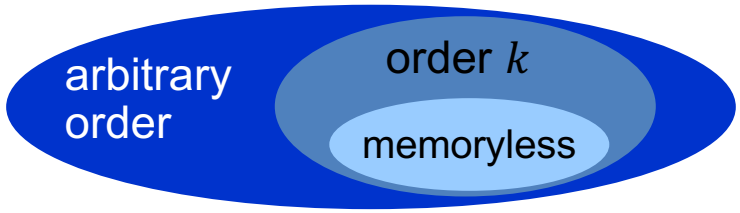
- *Universal data compression* deals with the optimal description of data in the absence of a given model
  - in most practical applications, the model is not given to us
- What is an "optimal description of x^n "?
  - There is always a binary code that assigns just 1 bit to the data at hand

$$C(y^n) = \begin{cases} 0 & y^n = x^n \\ 1 \text{ binary}(y^n) & y^n \neq x^n \end{cases}, \quad y^n \in A^n$$

  - To decode, the decoder would need to know x^n in advance, so why encode in the first place. This "optimal code for x^n " is meaningless.
  - We need a more meaningful notion of optimality.

Model Classes and Universal Coding

- *The answer: Model Classes*
  - *Examples:* all memoryless assignments,
    all Markov assignments of order k,
    all Markov assignments of arbitrary order.



- We want a "universal" code to perform as well as the best model in a *given* class $\mathcal{C}$ for *any* string x^n, where the best competing model changes from string to string. *Universality makes sense only w.r.t. a model class.*
- A code with length function $L(x^n)$ is *pointwise universal* w.r.t. a class $\mathcal{C}$ if

$$R_{\mathcal{C}}(L, x^n) = n^{-1}\left[L(x^n) - \min_{C' \in \mathcal{C}} L_{C'}(x^n)\right] \to 0 \quad \text{when } n \to \infty$$

pointwise redundancy

Example: Bernoulli Models

$$A = \{0,1\}, \qquad \mathcal{C} = \{P_\theta\ ,\ \theta \in \Theta = [0,1]\ \}$$

i.i.d. distribution with parameter $\theta = p(1)$

$$\min_{C \in \mathcal{C}} L_C(x^n) = \min_{\theta \in \Theta} \log \frac{1}{P_\theta(x^n)} = \log \frac{1}{P_{\hat{\theta}(x^n)}(x^n)} = n\,\hat{H}(x^n)$$

ML-estimate of θ

Therefore, our goal is to find a code such that

$$\frac{L(x^n)}{n} - \hat{H}(x^n) \xrightarrow{n \to \infty} 0$$

- Simple example of a universal code:
  Use $\lceil \log (n+1) \rceil$ bits to encode n_1 (maybe also n), then use an AC for $\hat{\theta} = \frac{n_1}{n}$, precisely the ML-estimate of θ

  $\Rightarrow$ *describe the best model to the decoder, then encode with that model*
- Code length $L(x^n) = n\hat{H}(x^n) + O(\log n)$: *goal met, but with two-part code, two-passes over the data (one to find n_1, one to encode).*

One-pass sequential schemes

- How about a one-pass, sequential version of this?
  Naïve *plug-in* approach:
  - after seeing $x_1, x_2, \ldots, x_t,$ encode x_{t+1} using $P(1|x^t) = \frac{n_1(x^t)}{t}$ (with an AC)
  - Not good! If $n_1 = 0$ or $n_1 = t$, we have events with $P = 0 \Longrightarrow -\log P = \infty$!!!
- *Laplace's rule of succession*: $P(1|x^t) = \frac{n_1(x^t)+1}{t+2}$
  - symbol probabilities always nonzero (initially 1/2, 1/2)
  - generalizes to $P(a|x^t) = \frac{n_a(x^t)+1}{t+\alpha}$ for $\alpha \geq 2$.
- *Krichevskii-Trofimov (KT) estimator*: $P(1|x^t) = \frac{n_1(x^t)+1/2}{t+1}$
  - generalizes to $P(a|x^t) = \frac{n_a(x^t)+1/2}{t+\alpha/2}$ for $\alpha \geq 2$.
  - code length $L(x^n) = n\hat{H}(x^n) + \frac{\alpha-1}{2}\log n + O(1)$

  best possible *pointwise* redundancy rate
- Both can be interpreted as *mixture models*

  $P(x^n) = \int_\theta W(\theta) P_\theta(x^n) d\theta$, for some *prior* $W(\theta)$

Sequential coding for Markov models

- Accumulate state-conditioned counts, use KT estimator *per-state*

Code length: $L(x^n|S) = n\hat{H}(x^n|S) + \frac{|S|(\alpha-1)}{2}\log n + O(1)$

state set

half the number of free statistical parameters

  - For k-th order Markov, $S = A^k$.
  - Applies also to FSM and tree sources, where S may be different.

- *Example:* binary Markov, $k = 1$, state set $S = \{0, 1\}$, initial state 0

x^{20} =	0	0	1	1	1	1	0	1	1	1	1	1	1	0	0	0	0	1	0	0
	1/2	3/4	1/6	1/2	3/4	5/6	1/8	3/8	7/10	3/4	11/14	13/16	5/6	3/20	1/2	7/12	9/14	5/16	5/22	13/18

Total probability:

$$Q(x^n) = \prod_{s\in S} \frac{\Gamma(n(0|s)+0.5)\Gamma(n(1|s)+0.5)}{n(s)!\,\Gamma(0.5)^2} = \frac{\Gamma(6.5)\Gamma(3.5)\Gamma(3.5)\Gamma(8.5)}{9!\,11!\,\Gamma(0.5)^4}$$

  - $-\log Q(x^n) = 17.61 \;\Rightarrow$ code length = 18 bits
  - $n\hat{H}(\mathrm{x}^{20}|S) = 9h\left(\frac{1}{3}\right) + 11h\left(\frac{3}{11}\right) \approx 17.56$ bits

Optimality and convergence rates

Normalized code length:

$$\frac{1}{n}L(x^n|S) = \underbrace{\hat{H}(x^n|S)}_{\to H(X) \text{ a. s. as } n\to\infty} + \underbrace{\frac{|S|(\alpha-1)}{2}\frac{\log n}{n} + \mathrm{O}(n^{-1})}_{\to 0 \text{ as } n\to\infty}$$

$\to H(X)$ a. s. as $n \to \infty$ if $X = X^\infty$ was indeed generated by the structure S

- In a stochastic setting, this code length is asymptotically optimal
  - how good is the convergence rate?

Theorem [Rissanen's lower bound] Let $\mathrm{C} = \{ P_\theta \mid \theta \in \Theta_K \}$ be a class of parametric models, where Θ_K is an open, bounded subset of $\mathcal{R}^K$. Assume that either the CLT holds for ML estimators of parameters in Θ_K, or

$$\Pr\{\sqrt{n}(\hat{\theta}(x^n) - \theta_i) \geq \log n\} \leq \delta(n) \to 0 \text{ as } n \to \infty.$$

Then for all Q and all $\varepsilon > 0$,

$$-n^{-1}E_\theta[\log Q(x^n)] \geq H_n(P_\theta) + \frac{K}{2}\frac{\log n}{n}(1-\varepsilon)$$

for all points in $\theta \in \Theta_K$ except in a set whose volume $\to 0$ as $n \to \infty$.

Optimality and convergence rates (cont.)

- This lower bound parallels Shannon's coding theorem:
  when we consider a model class instead of a single distribution,
  a *model cost* gets added to the entropy
  - Interpretation: if the parameters can be estimated well, they are "distinguishable" (P_θ is sensitive to θ), so the class cannot be coded without a model cost
  - The bound cannot hold for all models in the family, but it holds for most
- Conclusion: the number of parameters affects the achievable convergence rate of a universal code length to the entropy

$$\textit{Total length} = \underbrace{\text{Length}(\textit{model description})}_{\textit{model cost}} + \underbrace{\text{Length}(\textit{encoding}\,|\,\textit{model})}_{\textit{coding cost}}$$

model cost
proportional to the number of free statistical parameters

- *model cost = "cost of learning"*

Minimum Description Length (MDL) principle: the model that best describes the data is the model that minimizes the sum of the two parts.

Intuitive interpretations of model cost

- *Sparse statistics*
  - Many parameters to estimate $\Rightarrow$ samples will be "thinly spread"
    $\Rightarrow$ each parameter estimate will be based on few of samples
    $\Rightarrow$ *unreliable estimates*
  - k-th order Markov model over α symbols $\Rightarrow$ $\alpha^k(\alpha - 1)$ parameters
  - Example: $\alpha = 4, k = 10 \Rightarrow$ more than $3 \cdot 10^6$ parameters
  - Sometimes called *the curse of dimensionality*
- *There is a trade-off between how much the model "fits the data" and how reliable are the statistics collected to estimate the model*

Basic compressor/decompressor with context model

Compressor

Input: $x_1\ x_2\ x_3\ \ldots\ x_t\ \ldots$

Output: *binary stream*

1. Set $t = 1$, initialize
   $\text{count}(C) = 0,$
   $\text{count}(a|C) = 0, a \in A, C \in S$
2. Determine *context* C_t of x_t .
3. Retrieve $\hat{P}(\bullet|C_t),\ x_t$.
4. Encode x_t using $\hat{P}(\bullet|C_t)$.
5. Update $\hat{P}(\bullet|C_t)$:
   Increase $\text{count}(x_t|C_t)$ by 1.
   Increase $\text{count}(C_t)$ by 1.
   Done?
   No: $t \leftarrow t + 1$, go to 2.
   Yes: Stop.

Decompressor

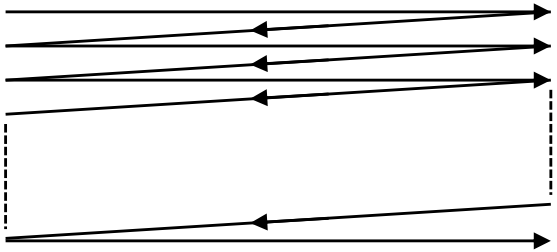
Input: *binary stream*

Output: $x_1\ x_2\ x_3\ \ldots\ x_t\ \ldots$

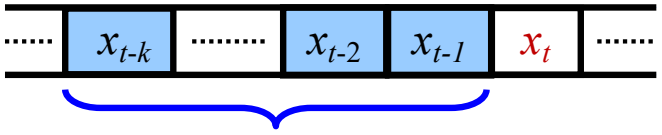
1. Set $t = 1$, initialize
   $\text{count}(C) = 0,$
   $\text{count}(a|C) = 0, a \in A, C \in S$
2. Determine *context* C_t of x_t .
3. Retrieve $\hat{P}(\bullet|C_t)$.
4. Decode x_t using $\hat{P}(\bullet|C_t)$.
5. Update $\hat{P}(\bullet|C_t)$:
   Increase $\text{count}(x_t|C_t)$ by 1.
   Increase $\text{count}(C_t)$ by 1.
   Done?
   No: $t \leftarrow t + 1$, go to 2.
   Yes: Stop.

Example: B/W image

- 1 b/w pixel = 1 bit
- Image size: 1800x2104 = 3,787,200 bits
- scanned in 1D raster order:



- modeled with k-th order binary Markov



k *"context"*

$\Longrightarrow 2^k$ parameters

- coded with KT probability assignment + AC

A Mathematical Theory of Communication

By C. E. SHANNON

INTRODUCTION

THE recent development of various methods of modulation such as PCM and PPM which exchange bandwidth for signal-to-noise ratio has intensified the interest in a general theory of communication. A basis for such a theory is contained in the important papers of Nyquist[1] and Hartley[2] on this subject. In the present paper we will extend the theory to include a number of new factors, in particular the effect of noise in the channel, and the savings possible due to the statistical structure of the original message and due to the nature of the final destination of the information.

The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point. Frequently the messages have *meaning*; that is they refer to or are correlated according to some system with certain physical or conceptual entities. These semantic aspects of communication are irrelevant to the engineering problem. The significant aspect is that the actual message is one *selected from a set* of possible messages. The system must be designed to operate for each possible selection, not just the one which will actually be chosen since this is unknown at the time of design.

If the number of messages in the set is finite then this number or any monotonic function of this number can be regarded as a measure of the information produced when one message is chosen from the set, all choices being equally likely. As was pointed out by Hartley the most natural choice is the logarithmic function. Although this definition must be generalized considerably when we consider the influence of the statistics of the message and when we have a continuous range of messages, we will in all cases use an essentially logarithmic measure.

The logarithmic measure is more convenient for various reasons:

1. It is practically more useful. Parameters of engineering importance

[1] Nyquist, H., "Certain Factors Affecting Telegraph Speed," *Bell System Technical Journal*, April 1924, p. 324; "Certain Topics in Telegraph Transmission Theory," *A. I. E. E. Trans.*, v. 47, April 1928, p. 617.

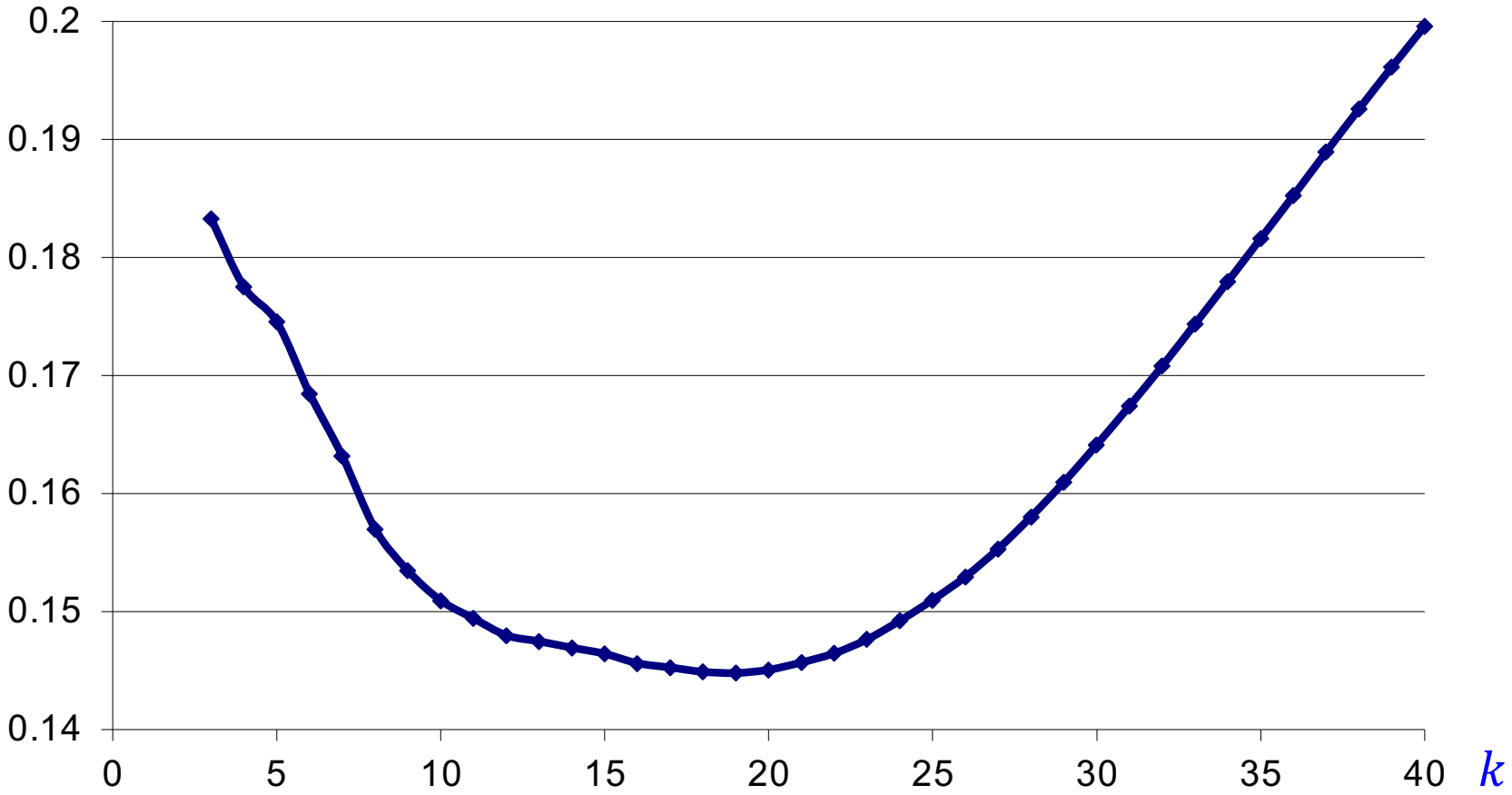
[2] Hartley, R. V. L., "Transmission of Information," *Bell System Technical Journal*, July 1928, p. 535.

Published in THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 27, pp. 379-423, 623-656, July, October, 1948

MONOGRAPH B-1598
Reissued December, 1957

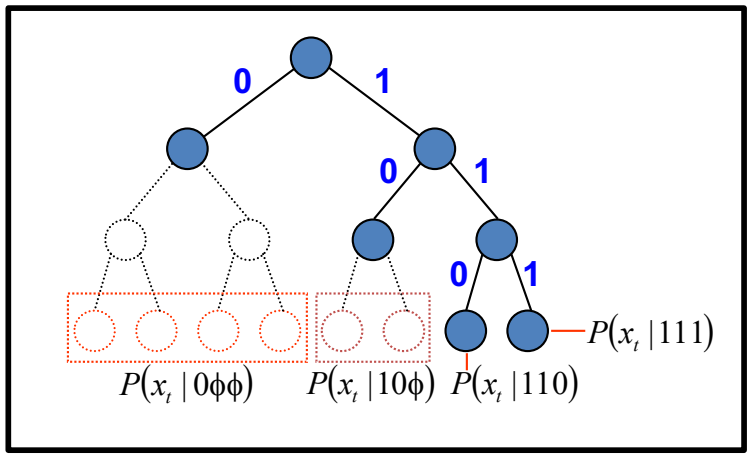
Compression ratio vs. Markov order

KT normalized code length (compression ratio)



How do we estimate the best context size?

- It depends on the computational environment and requirements
  - one way (suggested in previous example) is to try all possible values of k, find the one giving minimum code length, describe it in the output stream, and then encode *sequentially* using a model of that order. Decoder reads the value of k, and proceeds sequentially (statistics are *not* sent in the encoded stream).
    - asymmetric encoder/decoder complexity
  - *There are well known methods to do everything sequentially in one pass, or in two passes without exhaustive search, not just for fixed k, but also for tree sources.*



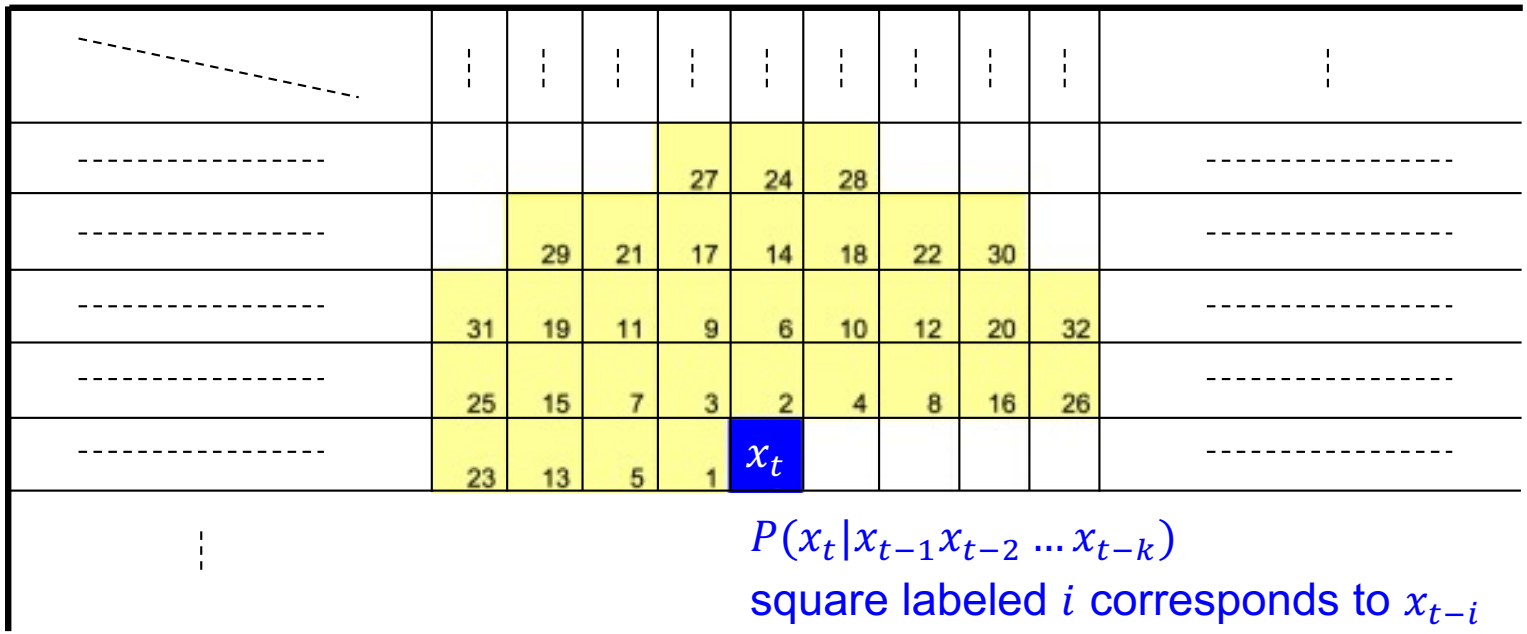
Tree example: 4 parameters instead of 8

- *"Context" algorithm*: [Rissanen'83], [Nohre'94], [Weinberger-Rissanen-Feder'95]
- *Context Tree Weighting (CTW)*: [Willems-Shtarkov-Tjalkens'95]
- *Linear time semi-predictive*: [Martín-Seroussi-Weinberger'04]

Two-dimensional contexts

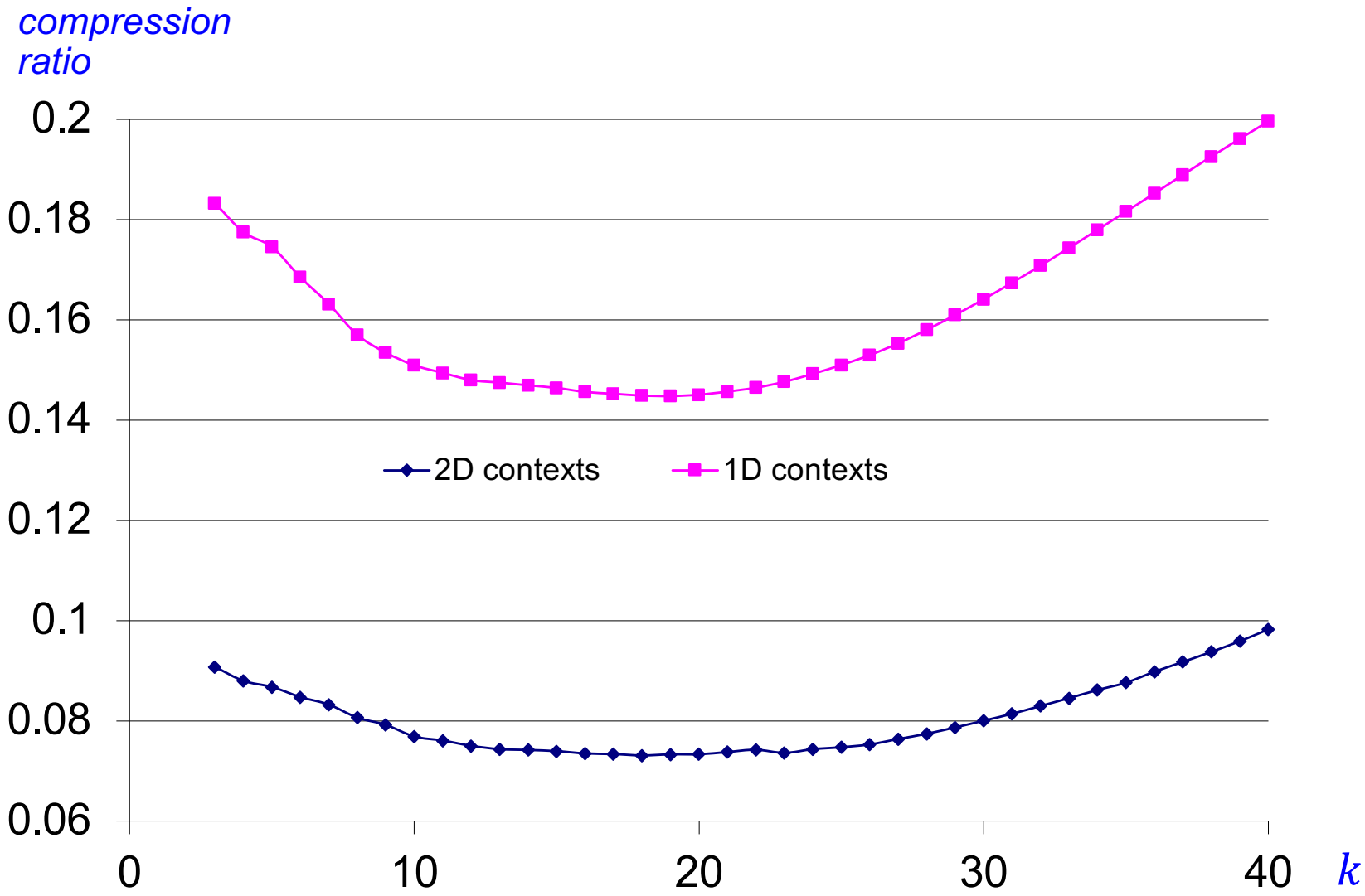
- Images are two-dimensional objects, so it makes sense to condition samples on their 2D neighborhoods
  - Example: order context samples by distance to modeled sample

Image

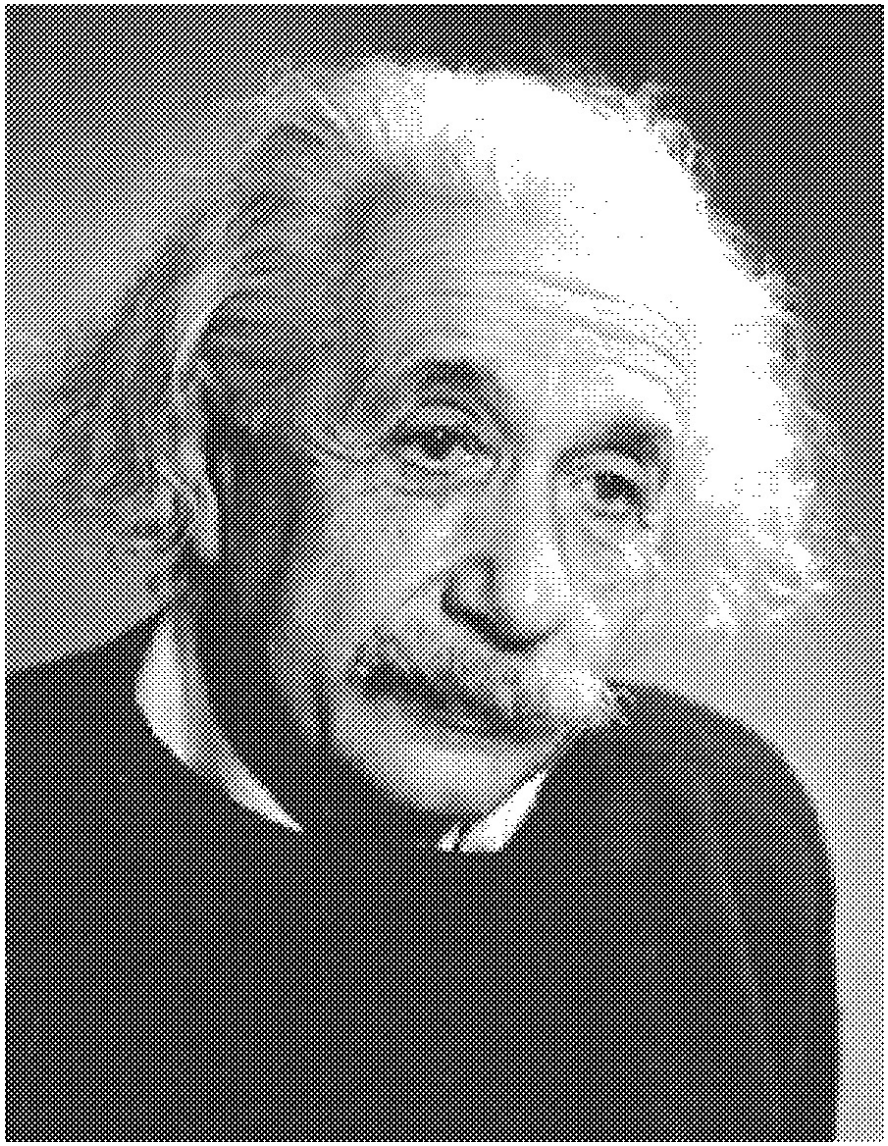


  - *causal conditioning*: dependence only on pixels *already encoded* (decoder can reconstruct the context)
  - not strictly a finite memory model, since full rows need to be stored in order to determine the next context
  - nevertheless, a finite-state model (FSM)

Compression ratio vs. 2D context size



Another B/W example



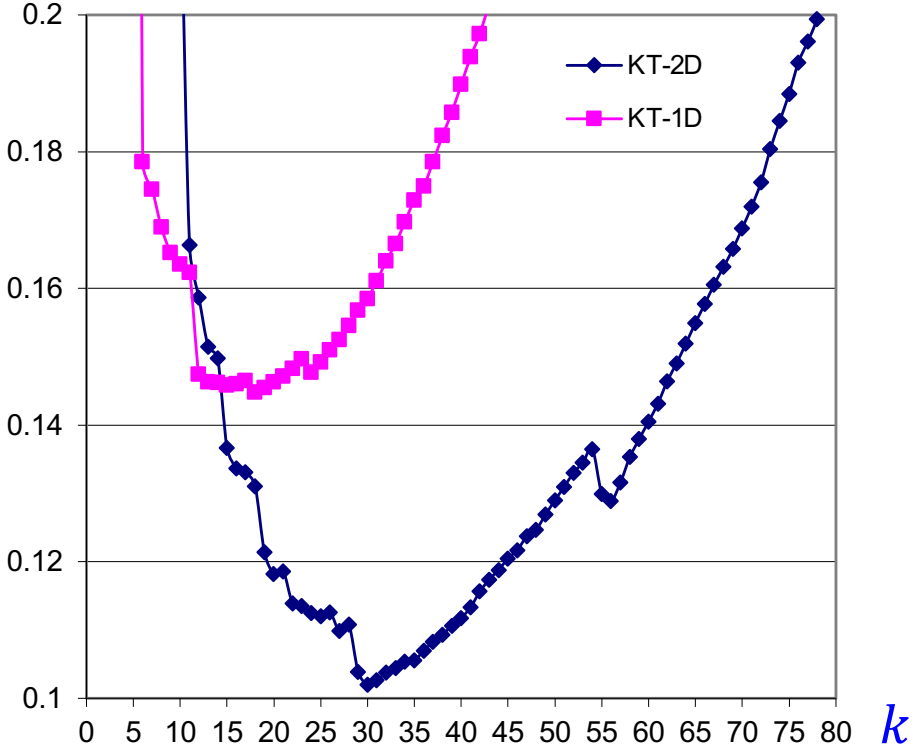
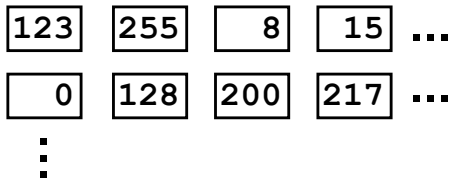


Image size: 896x1160

Going from binary images to grayscale images

- Everything seems to work quite well for binary images. How about more "natural" images?
- *Continuous tone images*
  - *Grayscale*: 2D array of *pixel intensity values* (integers) in a given range $[0..(\alpha - 1)]$ (often $\alpha = 256$)



  - *Color*: a 2D array of *vectors* (e.g. triplets) whose coordinates represent intensity in a given *color space* (e.g., RGB, YUV); similar principles
    - Alternative interpretation: a vector of grayscale images, one per color component (e.g., R, G, B).

- Even with almost minimal 2D context (3 closest samples, grayscale), we have $2^{24} \approx 17 \cdot 10^6$ possible different context patterns
  - gets astronomical very quickly: *big numbers = big problem*