LECTURE 4
COMPRESSION: ARITHMETIC CODING

Information Theory

Thomas Debris-Alazard

Inria, École Polytechnique

Huffman algorithm is optimal when compressing symbols of the alphabet $\mathcal{X}$

$\longrightarrow$ Don't forget: a memory-cost $O(\sharp\mathcal{X})$ to store the underlying tree

Though optimal, Huffman coding has issues

Huffman algorithm is optimal when compressing symbols of the alphabet $\mathcal{X}$

$\longrightarrow$ Don't forget: a memory-cost $O(\sharp\mathcal{X})$ to store the underlying tree

Though optimal, Huffman coding has issues

- If L successive outputs of X are independent, and if compressing successively to avoid memory issues, then average-length will belong to

$$\left[LH(\mathrm{X}), LH(\mathrm{X}) + L\right) \qquad \left(H\left(\mathrm{X}^{\otimes L}\right) = LH(\mathrm{X})\right)$$

$\longrightarrow$ It may happen that the $+L$ term is an overkill! $\Big($see Slide 28$\Big)$

Huffman algorithm is optimal when compressing symbols of the alphabet $\mathcal{X}$

$\longrightarrow$ Don't forget: a memory-cost $O(\sharp\mathcal{X})$ to store the underlying tree

Though optimal, Huffman coding has issues

- If L successive outputs of X are independent, and if compressing successively to avoid memory issues, then average-length will belong to

$$\left[LH(\mathsf{X}), LH(\mathsf{X}) + L\right) \qquad \left(H\left(\mathsf{X}^{\otimes L}\right) = LH(\mathsf{X})\right)$$

$\longrightarrow$ It may happen that the $+L$ term is an overkill! (see Slide 28)

- If L successive outputs of X are dependent, then
  - we loose a lot by compressing successively symbols, example of the language

    $\longrightarrow$ The last `z`-letter of `buzz` asks a lot while we know it will be `z` after reading `buz`
  - we could pack symbols within blocks of size L large enough

    $\longrightarrow$ But memory cost becomes $O(\sharp\mathcal{X}^L)$ ...

MOTIVATION

Huffman algorithm is optimal when compressing symbols of the alphabet $\mathcal{X}$

$\longrightarrow$ Don't forget: a memory-cost $O(\sharp\mathcal{X})$ to store the underlying tree

Though optimal, Huffman coding has issues

- If L successive outputs of X are independent, and if compressing successively to avoid memory issues, then average-length will belong to

$$\left[LH(\mathsf{X}), LH(\mathsf{X}) + L\right) \qquad \left(H\left(\mathsf{X}^{\otimes L}\right) = LH(\mathsf{X})\right)$$

$\longrightarrow$ It may happen that the $+L$ term is an overkill! (see Slide 28)

- If L successive outputs of X are dependent, then
  - we loose a lot by compressing successively symbols, example of the language

    $\longrightarrow$ The last z-letter of **buzz** asks a lot while we know it will be **z** after reading **buz**
  - we could pack symbols within blocks of size L large enough

    $\longrightarrow$ But memory cost becomes $O(\sharp\mathcal{X}^{L})\ldots$

We need an alternative to Huffman coding!

THE OBJECTIVE OF THE DAY

To present **arithmetic coding** and to **implement it**!

$\longrightarrow$ To understand where it is coming from we will come back “to the origin” of compression

COURSE OUTLINE

1. Some Reminders: Huffman Coding and Shannon Source Coding Theorem
2. Coding with Intervals: Shannon-Fano-Elias and Shannon
3. Arithmetic Coding

HUFFMAN AND SHANNON SOURCE CODING

Coding: use a table indexed by the letters to compress

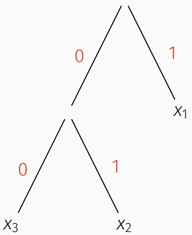
Decoding: use the associated binary tree

- Start at the root
- For each bit turn left or right
- When a leaf is attained, the corresponding letter is output and go back to the root

Consider

$\mathcal{X}$	$\varphi(x_i)$
x_1	1
x_2	01
x_3	00

, then



Be careful:

For tree representation: need to use a prefix code, otherwise codewords could be on a edge (decoding fails)

Expected length:

Given a distribution of symbols $\mathsf{X} : \Omega \to \{0,1\}^+$, the expected length of a symbol code $\varphi : \mathcal{X} \to \{0,1\}^+$ is

$$L(\varphi, \mathcal{X}) \stackrel{\text{def}}{=} \sum_{x \in \mathcal{X}} \ell(x)\ p(x) \quad \text{where } p(x) \stackrel{\text{def}}{=} \mathbb{P}(\mathsf{X} = x)$$

$\longrightarrow$ Expected length: measure of efficiency! We want it to be small

Optimal code:

A uniquely decodable code of a source X is optimal if there exists no other uniquely decodable code with a smaller expected length

Proposition: optimality of Huffman code

The Huffman code φ_{H} is a prefix code and it is optimal. Furthermore,

$$L\left(\varphi_{\mathrm{H}}, \mathcal{X}\right) \in [H(\mathsf{X}), H(\mathsf{X}) + 1)$$

Be careful: to build the Huffman code

- we need to know the distribution of the source $\mathsf{X} : \Omega \to \mathcal{X}$
- it requires $O(\sharp\mathcal{X})$ memory
- when compressing ≥ 2 letters in $\mathcal{X}$, we don't take into account the possible dependences

- Need to know beforehand the source statistics: wrong probability distribution q instead of p:

$$H(\mathrm{X}) + D_{\mathrm{KL}}(p||q) \leq L(\mathcal{X}, \varphi_{\mathrm{Huff}}) \leq H(\mathrm{X}) + D_{\mathrm{KL}}(p||q) + 1$$

$\Big($can be dealt with adaptative algorithm adapting statistics on the fly during the coding$\Big)$

- Based on a memoryless source model
- To overcome the issue with non-memoryless channel: pack letters in block of size L $\Big($realistic if L large enough$\Big)$

Coding efficiency $\frac{L(\varphi_{\mathrm{Huff}}, \mathcal{X}^L)}{H(\mathrm{X}_1, \ldots, \mathrm{X}_L)} \xrightarrow[L \to +\infty]{} 1$. But memory complexity $O\left(\sharp \mathcal{X}^L\right)$

How could we avoid these issues? Let us come back to Shannon source coding theorem for symbol codes...

Shannon's source coding theorem for symbol codes:

For any distribution $X : \Omega \to \mathcal{X}$, there exists a prefix code φ with expected length satisfying

$$L(\varphi, \mathcal{X}) < H(X) + 1$$

Furthermore, for any prefix code,

$$H(X) \leq L(\varphi, \mathcal{X})$$

To prove the upper-bound $\big($existential result$\big)$, we used code-lengths

$$\ell(x) \stackrel{\text{def}}{=} \lceil \log_2 1/p(x) \rceil$$

Historically it was the first idea to design compression scheme! Huffman thought differently

Historical ideas have an advantage: it leads to ***arithmetic coding***

Historical idea for compression:

Designing a scheme such that $\ell(x)$ is a

close as possible to $\log_2 1/p(x)$ to ensure $L(\varphi, \mathcal{X})$ as close as possible to $H(\mathsf{X})$

as in proof of Shannon's theorem!

INTERVAL CODING

Any numbers in $x \in [0,1]$ can be written as,

$$\sum_{i\geq 0} \ell_i 2^{-i} \quad \text{where } \ell_i \in \{0,1\}$$

We write $0.d_1 d_2 \ldots$

For any $\ell > 0$ and $x \in [0,1)$,

$$D_\ell(x) = (d_1, \ldots, d_\ell) \quad \text{denotes its first } \ell\text{-bits in its binary decomposition}$$

For instance:

0.25	$\rightarrow$	0.01	0.43	$\rightarrow$	$0.0110111000\ldots$
0.125	$\rightarrow$	0.001	0.71	$\rightarrow$	$0.1011010111\ldots$
0.625	$\rightarrow$	0.101	$1/\sqrt{2}$	$\rightarrow$	$0.1011010100\ldots$

Be careful:

Some numbers have many 2-adic representations, *e.g.* $0.25 \rightarrow 0.01$ and $0.25 \rightarrow 0.00111\ldots$

$\longrightarrow$ We will restrict to finite representations!

2-ADIC NUMBERS

2-adic numbers:

All the $x \in [0, 1)$ for which it exists $\ell > 0$ such that

$$x = \sum_{i=1}^{\ell} \ell_i 2^{-i} \quad \text{where } \ell_i \in \{0, 1\}$$

An important property:

For any 2-adic numbers u, v, if $D_\ell(u) = D_\ell(v)$ for some ℓ, then

$$|u - v| < 2^{-\ell}$$

Or equivalently, if two 2-adic numbers u, v verify $|u - v| \geq 2^{-\ell}$, then $D_\ell(u) \neq D_\ell(v)$

Proof:

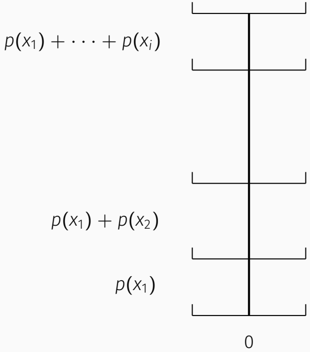
$$|u - v| = \left|\sum_{i=0}^{+\infty} (u_i - v_i) 2^{-i}\right| \leq \sum_{i=0}^{\infty} |u_i - v_i| 2^{-i} < \sum_{i=\ell+1}^{+\infty} 2^{-i} = 2^{-\ell}$$

This inequality is strict as $\forall i > \ell, |u_i - v_i| = 1$ implies that one of the sequence terminates by $111\ldots$: impossible as 2-adic numbers

SHANNON-FANO-ELIAS CODING: USE INTERVAL REPRESENTATION

$<$ a total order on $\mathcal{X}$

$$S(x) \stackrel{\text{def}}{=} \sum_{y<x} p(y)$$



The interval width is $p(x_i)$ and this width necessitates $\approx \log_2 p(x_i)$ bits

$\longrightarrow$ Each x_i is encoded by its interval given by $p(x_i)$

$$\overline{S}(x) \stackrel{\text{def}}{=} \frac{p(x)}{2} + \sum_{y<x} p(y)$$

$\longrightarrow \overline{S}(x)$ is the middle of the interval $\left[\sum_{y<x} p(y), \sum_{y\leq x} p(y)\right]$

Shannon-Fano-Elias coding:

Let $\varphi_{\text{SFE}}(x) \stackrel{\text{def}}{=} D_{d(x)+1}\left(\overline{S}(x)\right)$ where $d(x) \stackrel{\text{def}}{=} \lceil \log_2 1/p(x) \rceil$. Then φ_{SFE} is a coding

- prefix
- $L(\varphi_{\text{SFE}}, \mathcal{X}) < H(\mathsf{X}) + 2$

x_i	$p(x_i)$	$\lceil \log_2 1/p(x_i) \rceil$	$\overline{S}(x_i)$		$\varphi_{\text{SFE}}(x_i)$	$\varphi_{\text{Huff}}(x)$
a	0.43	3	0.215	0.0011011 . . .	001	0
b	0.17	4	0.515	0.1000001 . . .	1000	100
c	0.15	4	0.675	0.1010110 . . .	1010	101
d	0.11	5	0.805	0.1100111 . . .	11001	110
e	0.09	5	0.905	0.1110011 . . .	11100	1110
f	0.05	6	0.975	0.1111100 . . .	111110	1111

$\longrightarrow$ Sannon-Fano-Elias coding is not optimal. . .

Shannon-Fano-Elias Coding is Prefix

$$\overline{S}(x) \stackrel{\text{def}}{=} \frac{p(x)}{2} + \sum_{y<x} p(y), \quad d(x) = \lceil \log_2 1/p(x) \rceil \quad \text{and} \quad \varphi_{\text{SFE}}(x) = D_{d(x)+1}\left(\overline{S}(x)\right)$$

Proof: φ_{SFE} is prefix

Let $x \neq y$ such that $\ell(x) \leq \ell(y)$,

- If $x < y$,

$$\begin{aligned}
\overline{S}(y) - \overline{S}(x) &= \frac{p(y)}{2} + \sum_{z<y} p(z) - \frac{p(x)}{2} - \sum_{z<x} p(z) \\
&= \frac{p(y)}{2} - \frac{p(x)}{2} + \sum_{x \leq z < y} p(z) \\
&= \frac{p(y)}{2} + \frac{p(x)}{2} + \sum_{x < z < y} p(z) \\
&> \max\left(\frac{p(y)}{2}, \frac{p(x)}{2}\right) \\
&\geq \max\left(2^{-d(x)-1}, 2^{-d(y)-1}\right) \\
&= 2^{-\ell(x)-1}
\end{aligned}$$

- Same result for if $y < x$

Therefore, $\overline{S}(y)$, $\overline{S}(x)$ differs on their first $(d(x)+1)$-bits and $\varphi_{\text{SFE}}(x)$ cannot be a prefix of $\varphi_{\text{SFE}}(y)$

$$\overline{S}(x) \stackrel{\text{def}}{=} \frac{p(x)}{2} + \sum_{y<x} p(y), \quad d(x) = \lceil \log_2 1/p(x) \rceil \quad \text{and} \quad \varphi_{\text{SFE}}(x) = D_{d(x)+1}\left(\overline{S}(x)\right)$$

Proof: $L(\varphi_{\text{SFE}}, \mathcal{X})$

By definition

$$\begin{aligned} L(\varphi_{\text{SFE}}, \mathcal{X}) &= \sum_x p(x)\ell(x) \\ &= \sum_x p(x)(\lceil \log_2 1/p(x) \rceil + 1) \\ &\leq \sum_x p(x) \log_2(1/p(x)) + 2p(x) \\ &= H(\mathsf{X}) + 2 \end{aligned}$$

Shannon coding: defined as Shannon-Fano-Elias coding, ***at the exception:***

- The symbols are ordered by their decreasing probability, *i.e.*, $x < y \iff p(x) \geq p(y)$.
- We compress x with the first $d(x) \stackrel{\text{def}}{=} \lceil -\log_2 p(x) \rceil$ bits of $S(x) \stackrel{\text{def}}{=} \sum_{y<x} p(y)$

Shannon Coding:

Let $\varphi_{\text{Sh}}(x) \stackrel{\text{def}}{=} D_{d(x)}\Big(S(x)\Big)$ where $d(x) \stackrel{\text{def}}{=} \lceil \log_2 1/p(x) \rceil$ and $S(x) \stackrel{\text{def}}{=} \sum_{y<x} p(y)$. Then φ_{Sh} is a coding

- prefix
- $L(\varphi_{\text{Sh}}, \mathcal{X}) < H(X) + 1$

$$\Big(\text{Notice that } \ell(x) = d(x) \text{ in Shannon coding} \Big)$$

$$S(x) \stackrel{\text{def}}{=} \textstyle\sum_{y<x} p(y), \quad d(x) = \lceil \log_2 1/p(x) \rceil \quad \text{and} \quad \varphi_{\mathrm{Sh}}(x) = D_{d(x)}(S(x))$$

Proof: φ_{Sh} is prefix

Let $x \neq y$ with $\ell(x) = \lceil \log_2 1/p(x) \rceil \leq \ell(y) = \lceil \log_2 1/p(y) \rceil \iff p(x) \geq p(y) \iff x < y$.

$$\begin{aligned} S(y) - S(x) &= \sum_{z<y} p(z) - \sum_{z<x} p(z) \\ &= \sum_{x \leq z < y} p(z) \\ &= p(x) + \sum_{x<z<y} p(z) \\ &\geq p(x) \\ &\geq 2^{-\ell(x)} \end{aligned}$$

Therefore, $S(y)$, $S(x)$ differs on their first $\ell(x)$-bits and $\varphi_{\mathrm{Sh}}(x)$ cannot be a prefix of $\varphi_{\mathrm{Sh}}(y)$

$$S(x) \stackrel{\text{def}}{=} \sum_{y<x} p(y), \quad d(x) = \lceil \log_2 1/p(x) \rceil \quad \text{and} \quad \varphi_{\text{Sh}}(x) = D_{d(x)}(S(x))$$

Proof: $L(\varphi_{\text{Sh}}, \mathcal{X})$

By definition,

$$\begin{aligned} L(\varphi_{\text{Sh}}, \mathcal{X}) &= \sum_x p(x)\ell(x) \\ &= \sum_x p(x)(\lceil \log_2 1/p(x) \rceil) \\ &\leq \sum_x p(x) \log_2(1/p(x)) + 2p(x) \\ &= H(\mathsf{X}) + 1 \end{aligned}$$

Dyadic case: $\forall x$, $p(x) = 2^{-j(x)}$ which implies that $p(x) = 2^{-\lceil \log_2 p(x) \rceil}$

In this case: lengths of encoding satisfy: $\ell(x) = -\lceil \log_2 p(x) \rceil = -\log_2 p(x)$ and the average,

$$L(\varphi_{\mathrm{Sh}}, \mathcal{X}) = H(\mathsf{X})$$

which is optimal by Shannon source coding theorem!

Consider the following source: a with probability $1 - 2^{-10}$ and b with probability 2^{-10}

$$|\varphi_{\mathrm{Sh}}(\mathrm{a})| = 10 \quad \text{and} \quad |\varphi_{\mathrm{Sh}}(\mathrm{b})| = \lceil -\log_2(1 - 2^{-10}) \rceil = \lceil 0.0014 \rceil = 1$$

But Huffman coding uses 1 bit to compress a and 1 bit to compress b

$\longrightarrow$ Shannon's code is not optimal!

ARITHMETIC CODING

- Main drawback of Huffman: compressing blocks of size L has memory cost $\mathrm{Mem} = O\left(\sharp \mathcal{X}^L\right)$ and,

$$\text{efficiency} = \frac{L(\varphi_{\text{Huff}}, \mathcal{X}^L)}{H(\mathrm{X}^{\otimes L})} \approx \frac{H\left(\mathrm{X}^{\otimes L}\right) + 1}{H\left(\mathrm{X}^{\otimes L}\right)} \approx 1 + \frac{1}{LH(\mathrm{X})} = 1 + O\left(\frac{1}{L}\right) = 1 + O\left(\frac{1}{\log \mathrm{Mem}}\right)$$

$\longrightarrow$ We need Mem to be huge $\left(L = \log \text{Mem to be large}\right)$ for an efficiency ≈ 1

- Arithmetic coding allows to work on blocks of arbitrary sizes L with acceptable algorithmic cost depending of the distribution model. Furthermore, efficiency remains the same

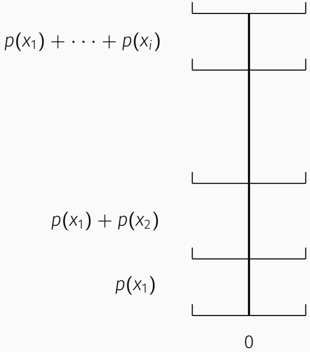
$$\text{efficiency} = 1 + O\left(\frac{1}{L}\right)$$

$\longrightarrow$ But it relies on the ability to perform computations with arbitrary large precision!

FUNDAMENTAL IDEA (I)

$<$ a total order on $\mathcal{X}$

$$S(x) \stackrel{\text{def}}{=} \sum_{y<x} p(y)$$



The interval width is $p(x_i)$ and this width necessitates $\approx \log_2 p(x_i)$ bits

$\longrightarrow$ Fundamental idea: to encode x_i identify the interval coming from $+p(x_i)$ with its length

Instead of encoding symbols of $\mathcal{X}$, **work directly on $\mathcal{X}^L$** equipped with lexicographical order

To encode $(x_1, \ldots, x_L)$,

1. Compute the interval

$$\Big[S(x_1, \ldots, x_L), S(x_1, \ldots, x_L) + p(x_1, \ldots, x_L)\Big]$$

$$\text{where } S(x_1, \ldots, x_L) \stackrel{\text{def}}{=} \sum_{(y_1, \ldots, y_L) < (x_1, \ldots, x_L)} p(y_1, \ldots, y_L)$$

2. Encode $(x_1, \ldots, x_L)$ with an element of the interval whose 2-adic representation length is

$$\lceil \log_2 p(x_1, \ldots, x_L) \rceil \quad \Big(\text{more precisely, we can use} \leq \log_2 p(x_1, \ldots, x_L) + 2 \text{ bits}\Big)$$

Identifying the interval $\Longrightarrow$ deduce $x_1, \ldots, x_L$ at the decoding step!

$$|\varphi_{AC}(x_1, \ldots, x_L)| = -\log_2 p(x_1, \ldots, x_L) + O(1)$$

Therefore,

$$L(\varphi_{AC}, \mathcal{X}^L) = H(X_1, \ldots, X_L)\, L + O(1)$$

$\longrightarrow$ $O(1)$ additional bits are wasted to encode L symbols and not $O(L)$ as with L-repetitions of Huffman!

When the $+1$ in Huffman coding is an overkill:

Consider the source: a with probability $1 - 2^{-10}$ and b with probability 2^{-10},

$$L\left(\varphi_H, \{a, b\}\right) = 1$$

Consider L independent outputs of the source and compress them successively with Huffman:

$$\forall (x_1, \ldots, x_L) \in \mathcal{X}^L, \quad |\varphi_H(x_1, \ldots, x_L)| = L$$

Arithmetic coding outperforms Huffman:

$$L\left(\varphi_{AC}, \{a, b\}^L\right) = h\left(2^{-10}\right) L + O(1) \approx 0.11L + O(1)$$

Given the order $<$ on $\mathcal{X}$, we have the lexicographical order on $\mathcal{X}^L$

Issue:

We need to compute the interval,

$$\Big[S(x_1,\dots,x_L), S(x_1,\dots,x_L)+p(x_1,\dots,x_L)\Big]$$

$$\text{where } S(x_1,\dots,x_L) \stackrel{\text{def}}{=} \sum_{(y_1,\dots,y_L)<(x_1,\dots,x_L)} p(y_1,\dots,y_L)$$

with a precision of $\lceil \log_2 p(x_1,\dots,x_L) \rceil$-bits!

$\longrightarrow$ *To implement arithmetic code: need to perform computations with arbitrary large precision...*

By supposing that we can perform computation with arbitrary length, we also need to compute efficiently

$$\sum_{(y_1,\ldots,y_L)<(x_1,\ldots,x_L)} p(y_1,\ldots,y_L) \quad \text{and} \quad p(x_1,\ldots,x_L)$$

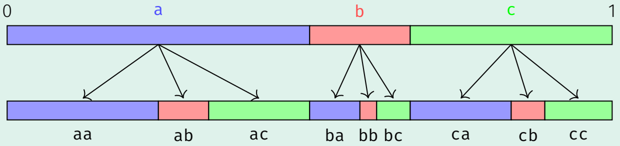
Instead of encoding symbols of $\mathcal{X}$, work directly on $\mathcal{X}^L$ by identifying $(x_1, \ldots, x_L)$ to a sequence of sub-intervals decomposed via the conditional probabilities

$$p(x_1) \to p(x_1)p(x_1 \mid x_2) \to (p(x_1)p(x_1 \mid x_2))\, p(x_3 \mid x_2, x_1) \to \cdots$$

$\longrightarrow$ We can identify iteratively the inverval

An example: a fractal phenomenon when the outputs are independent

Suppose $p(\mathrm{a}) = \frac{1}{2}$, $p(\mathrm{b}) = \frac{1}{6}$ and $p(\mathrm{c}) = \frac{1}{3}$,



$\mathcal{X} = \{a_1, \ldots, a_K, \perp\}$ where $\perp$ is a special symbol notifying the "end of file"

$\Big($ before we fixed an order $<$ on $\mathcal{X}$, it is here implicitly given by the indexes $\Big)$

We will work on $\mathcal{X}^L$ with lexicographical order

1. We divide the real line $[0, 1)$ into K intervals of lengths equal to the probabilities $\mathbb{P}(x_1 = a_i)$
2. Then, we take each interval a_i and subdivide it into intervals $a_i a_1, a_i a_2, \ldots, a_i a_K$ s.t $a_i a_j$ has length $p(x_2 = a_j \mid a_i)$ relatively to a_i,
$$p(x_1 = a_i)p(x_2 = a_j \mid x_1 = a_i) = p(x_1 = a_i, x_2 = a_j)$$
3. We iterate this procedure

Iterative procedure to find $[u, v)$ for the input string

Input: $(x_1, \ldots, x_L)$

$u := 0$

$v := 1$

$p := v - u$

`for` $n = 1$ to L {

Compute the cumulative probabilities Q_n and R_n

$v := u + pR_n(x_n \mid x_1, \ldots, x_{n-1})$

$u := u + pQ_n(x_n \mid x_1, \ldots, x_{n-1})$

$p := v - u$

}

$$Q_n(a_i \mid x_1, \ldots, x_{n-1}) \stackrel{\text{def}}{=} \sum_{K=1}^{i-1} p(x_n = a_K \mid x_1, \ldots, x_{n-1})$$

$$R_n(a_i \mid x_1, \ldots, x_{n-1}) \stackrel{\text{def}}{=} \sum_{K=1}^{i} p(x_n = a_K \mid x_1, \ldots, x_{n-1})$$

Encoding:

To encode $x_1, \ldots, x_N$ use the above procedure to compute $[u, v)$ and return a binary string whose interval lies within that interval

$$Q_n(a_i \mid x_1, \ldots, x_{n-1}) = \sum_{K=1}^{i-1} p(x_n = a_K \mid x_1, \ldots, x_{n-1})$$

$$R_n(a_i \mid x_1, \ldots, x_{n-1}) = \sum_{K=1}^{i} p(x_n = a_K \mid x_1, \ldots, x_{n-1})$$

Proposition:

If $p = p(x_1, \ldots, x_{n-1})$, then

$$pQ_n(x_n \mid x_1, \ldots, x_{n-1}) - pR_n(x_n \mid x_1, \ldots, x_{n-1}) = p(x_1, \ldots, x_n)$$

$\longrightarrow$ At step n, the width $[u, v)$ has length $p(x_1, \ldots, x_n)$

In the following proposition we use the lexicographical order on $\mathcal{X}^n$ given the order over

$\mathcal{X} = \{a_1, \ldots, a_I\}$ *with* $a_1 \leq \cdots \leq a_I$

Proposition:

If $p = p(x_1, \ldots, x_{n-1})$, $u = \sum_{(y_1,\ldots,y_{n-1})<(x_1,\ldots x_{n-1})} p(y_1, \ldots, y_{n-1})$ and

$v = \sum_{(y_1,\ldots,y_{n-1})\leq(x_1,\ldots x_{n-1})} p(y_1, \ldots, y_{n-1})$ then,

$$u + pQ_n = \sum_{(y_1,\ldots,y_n)<(x_1,\ldots x_n)} p(y_1, \ldots, y_n) \quad \text{and} \quad v + pR_n = \sum_{(y_1,\ldots,y_n)\leq(x_1,\ldots x_n)} p(y_1, \ldots, y_n)$$

$\longrightarrow$ At each step we are in the right interval

$r = \varphi_{\mathrm{AC}}(x_1, \ldots, x_L)$ be the real number whose 2-adic representation encodes $(x_1, \ldots, x_L) \in \mathcal{X}^L$.

The letters $x_1, \ldots, x_L$ are the only ones for which

$$
\begin{aligned}
\sum_{y_1 < x_1} p(y_1) \quad &< r < \quad \sum_{y \leq x_1} p(y_1) \\
\sum_{(y_1, y_2) < (x_1, x_2)} p(y_1, y_2) \quad &< r < \quad \sum_{(y_1, y_2) \leq (x_1, x_2)} p(y_1, y_2) \\
&\vdots \\
\sum_{(y_1, \ldots, y_n) < (x_1, \ldots x_n)} p(y_1, \ldots, y_n) \quad &< r < \quad \sum_{(y_1, \ldots, y_n) \leq (x_1, \ldots x_n)} p(y_1, \ldots, y_n)
\end{aligned}
$$

To communicate L letters: both the encoder and decoder need to compute $O(L\sharp\mathcal{X})$ conditional probabilities

In Huffman coding: all the $\sharp\mathcal{X}^L$-sequences have to be considered!

Be careful:

How can we compute the conditional probabilities? Do we need to store them?

- If the outputs source are independent: only need to store $p(a_1), \ldots, p(a_l)$
- Markov chain $\big($under the right hypotheses$\big)$: only need to store initial distribution and transition matrix

$\longrightarrow$ In arithmetic coding we need to model our data. It fits well with adaptative Bayesian model

Bayesian Model:

Probabilities are used both for uncertainty of outputs but also on the parameter of the model!

Suppose that Bill $\big($your best friend$\big)$, send N times a coin observing a sequence of n_h heads. We don't know the probability f_h of the coin to be head. But we know that Bob has previously chosen a coin such that f_h follows a known distribution. What is the probability that the $N+1$-outcome is head?

$$\mathcal{X} = \{\mathtt{a}, \mathtt{b}\}$$

$$p(\mathtt{a} \mid x_1, \ldots, x_{n-1}) = \frac{F_\mathtt{a} + 1}{F_\mathtt{a} + F_\mathtt{b} + 2}$$ where $F_\mathtt{x}$: number of times that $\mathtt{x}$ has occurred in $x_1, \ldots, x_{n-1}$

$\longrightarrow$ This “adaptative” model follows from simple assumptions!

Arithmetic Coding really appreciates this model to compute iteratively

$$Q_n(a_i \mid x_1, \ldots, x_{n-1}) = \sum_{k=1}^{i-1} p(x_n = a_k \mid x_1, \ldots, x_{n-1})$$

$$R_n(a_i \mid x_1, \ldots, x_{n-1}) = \sum_{k=1}^{i} p(x_n = a_k \mid x_1, \ldots, x_{n-1})$$

EXERCISE SESSION