UNIT-3

PART-1

ARITHMETIC CODE & its Applications

Introduction

→ Huffman coding guarantees a coding Rate R within 1 bit of Entropy H

→ It generates a code whose rate is within "P_{max} + 0.086 of Entropy"
where P_{max} is the Probability of Most frequently used symbols.

→ for a large Alphabet set the value of P_{max} is significantly small & Deviation from entropy is quite Small.

→ Now for those cases where alphabet is small & probabilites of letters are Skewed, than the value of P_{max} is quite large. and Huffman codes become inefficient when compared to Entropy.

→ Possible Solution (1)

Make a Block of more than one symbols & generate the Extended Huffman Code.
(This Approach doesn't work always)

Ex - $A = \{a_1, a_2, a_3\}$

$P(a_1) = 0.95$, $P(a_2) = 0.02$ & $P(a_3) = 0.03$.

The Entropy of the source = 0.335 bits/symbol
(Entropy shows the lowest rate at w/c source can code)

Symbols	$P(a_i)$	$C(a_i)$
a_1	0.95	0
a_2	0.02	11
a_3	0.03	10

The avg length = 1.05 bits/sym.

Difference b/w Avg length & Entropy = 0.715 bits/sym

w/c is 213% of Entropy.

This means in order to code this sequence.
we would need more than twice the no. of bits.
promised by Entropy

Slide 2. Arithmatic Coding.

Arithmatic Coding — 3 ways of Encoding.

→ generates variable length code

→ Useful when dealing with source with Small Alphabet such binary sources

→ a unique Identifies / tag → Repressed by a binary fraction value
block coding

Imp → Tag values [0,1]

lower limit = 0   upper limit = 1

Sequence $x = \{x_1, x_2 \ldots x_n\}$

$$l^n = l^{n-1} + (u^{n-1} - l^{n-1}) \times f_x(x_{n-1})$$

$$u^n = l^{n-1} + (u^{n-1} - l^{n-1}) \times f_x(x_n)$$

n → length of the sequence.

x_n → Sequence.

Ex. 4 5 6

$n=1$ $n=2$ $n=3$

$x_n = 4$ $x_n = 5$ $x_n = 6$

$$\boxed{T_x(x) = \frac{u^n + l^n}{2}}$$

~~Basic~~ Idea -

(1) Repeat each string x of length n by a unique Inter [$[L,U]$ in rang $[0,1]$.

(2) The width of ~~R-L~~ of Interval $[L,U]$ represents the probability of Occurring x.

(3) The Interval $[L,U]$ can itself be represented by any no. called Tag.

for 3rd Element -

$$l_B^3 = l^2 + (u^2 - l^2) \times f_x(2-1)$$
$$= 0.656 + (0.8 - 0.656) \times f_x(1)$$
$$= 0.656 + (0.8 - 0.656) \times 0.8$$
$$= 0.7712$$

$$u_B^3 = l^2 + (u^2 - l^2) \times f_x(2)$$
$$= 0.656 + (0.8 - 0.656) \times 0.82$$
$$= 0.77408$$

The tag is contained in the interval [0.7712, 0.77408]

for 4th Element: '1'

$$l^4 = l^3 + (u^3 - l^3) * f_x(1-1)$$
$$= 0.7712 + (0.77408 - 0.7712) \times f_x(0)$$
$$= 0.7712$$

$$u^4 = l^3 + (u^3 - l^3) \times f_x(1)$$
$$= 0.7712 + (0.77408 - 0.7712) \times 0.8$$
$$= 0.773504$$

The Tag is

$$T_x(1321) = \frac{u^4 + l^4}{2} = \frac{0.773504 + 0.7712}{2}$$
$$= \boxed{0.772352}$$

Consider a source with Alphabet $A = \{a_1, a_2, a_3\}$ with probability model $P(a_1) = 0.8$, $P(a_2) = 0.02$ & $P(a_3) = 0.18$. [$H(S) = 0.816$ bits/sym]

Eg Encode sequence "1 3 2 1" / $a_1\ a_3\ a_2\ a_1$

$F_X(k) = 0 \qquad k \le 0$

$F_X(1) = 0.8 \qquad F_X(2) = 0.82 \qquad F_X(3) = 1 \qquad F_X(k) = 1, \; k \ge 3$

Soln Initialize $l^0 = 0 \qquad u^0 = 1$

for Ist Element : 1

$$l^1 = l^0 + (u^0 - l^0) \times F_X(1-1)$$
$$= 0 + (1-0) \times F_X(0) = 0$$
$$u^1 = l^0 + (u^0 - l^0) \times F_X(1)$$
$$= 0 + (1-0) \times 0.8 = 0.8$$

$$T_X(1) = \frac{u^1 + l^1}{2} = \frac{0.8+0}{2} = 0.4.$$

The tag is contained in the interval = [0, 0.8]

for 2nd Element : 3

$$l^2 = l^1 + (u^1 - l^1) \times F_X(3-1)$$
$$= 0 + (0.8-0) \times F_X(2)$$
$$= 0 + 0.8 \times 0.82 = \cancel{0.82}\ 0.656$$

$$u^2 = l^1 + (u^1 - l^1) \times F_X(3)$$
$$= 0 + (0.8 - 0) \times 1 = 0.8.$$

The tag is contained in the interval = [0.656, 0.8]

Note: Generation of tag works by reducing the size of the interval in which the tag Resides as more as element of sequence are recieved.

Arithmatic

~~P(A) = 0.5 P(B) =~~

Procedure –

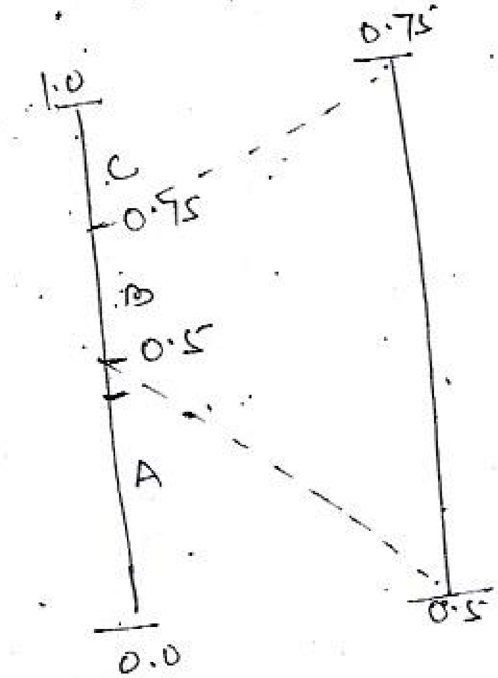
Step 1 – Divide the neumeric Range 0 to 1 into no. of different symbol Present in the message

Step 2. Expand the first letter to be coded Along with the Range further subdivided ~~into~~ this Range into no. of symbols.

Step 3. Repeat the procedure untill termination character is Encoded

Ex – P(A) = 0.5   P(B) = 0.25 & P(C) = 0.25

Sequence: BACA



A [0, 0.5]

B [0.5, 0.75]

Generation of Tag

→ we require cdf to map sequence of symbols in a unit interval.

$A = \{a_1, a_2 \cdots a_m\}$! $P(a_i)$ Random Variable

$X(a_i) = \underline{i}$

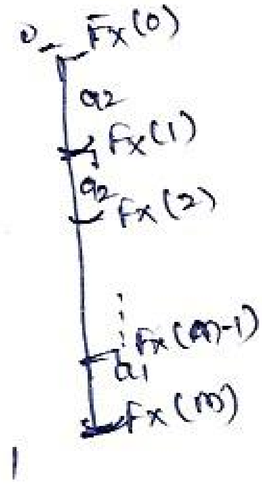
↑ index of sym.

PDF : $P(X=i) = P(a_i)$

CDF : $F_X(i) = \sum_{k=1}^{i} P(X=k)$

IDEA : Reduce the size of interval in w/c the tag resides as more elements of the sequence are received

→ Partition the unit Interval into sub-intervals of the form



$a_i \in [F_X(i-1), F_X(i))$

Algo

(1) Divide the unit interval into sub intervals of the form $[F_X(i-1), F_X(i))$, $i = 1, 2, \ldots m$

(2) Associate the sub intervals with symbols a_i

(3) ~~of~~ choose the interval according to the occurance of symbol in the sequence.

(3) The the interval contains the tag value will be sub interval $[F_X(K-1), F_X(K)]$

(4) Now ~~partition~~ so subinterval partitioned in exactly the same proportions as the original inter.

Decoding in Arithmatic Coding

Ex Decipher the Tag = "0.772352" [0,1] by 4 step

$F_X(1) = 0.8$, $F_X(2) = 0.82$, $F_X(3) = 1$, $F_X(k) = 0, k \le 0$

S1 Initialize $l^0 = 0$, $u^0 = 1$

$l^1 = l^0 + (u^0 - l^0) \times F_X(x_n - 1)$

$= 0 + (1-0) \times F_X(x_n - 1) = F_X(x_n - 1)$

$u^1 = l^0 + (u^0 - l^0) \times F_X(x_n)$

$= 0 + (1-0) \times F_X(x_n) = F_X(x_n)$

The Ist sequence lies $[F_X(x_n - 1), F_X(x_n))$

let $x_n = 1 \Rightarrow [0, 0.8]$

Is the tag lies b/w the

tag value lies b/w 0 & 0.8 when $x_n = 1$

sequence value = "1"

Step 2 $l^2 = l^1 + (u^1 - l^1) \times F_X(x_n - 1)$

$= 0 + (0.8 - 0) F_X(x_n - 1)$

$= 0.8 F_X(x_n - 1)$

$u^2 = l^1 + (u^1 - l^1) \times F_X(x_n)$

$= 0.8 * F_X(x_n)$

- II^{nd} sequence lies b/w $[0.8\,F_X(x_n-1),\ 0.8\,F_X(x_n)]$

let $x_n = 1 \Rightarrow [0, 0.64]$

Tag value doesn't lie b/w interval.

Now let $x_n = 2$ $\quad [0.64, 0.656]$

Again value doesn't lie interval,

Now let $x_n = 3$ $\quad [0.656, 0.8]$

tag values lies in range $[0.656, 0.8)$ when $x_n = 3$

So <u>sequence value → '3'</u>

<u>Step 3.</u> $l^3 = l^2 + (u^2 - l^2) \times F_X(x_n - 1)$

$= 0.656 + 0.144 \times F_X(x_n - 1)$

$u^3 = l^2 + (u^2 - l^2) \times F_X(x_n)$

$= 0.656 + 0.144 \times F_X(x_n)$

$x_n = 1 \Rightarrow [0.656, 0.7712]$ ×

$x_n = 2 \Rightarrow [0.7712, 0.77408]$ ✓

tag value

∴ sequence value → '2'

Step 4

$l^4 = 0.7712 + 0.00288 \times F_x(x_4 - 1)$

$u^4 = 0.7712 + 0.00288 \times F_x(x_4)$

$x_n = 1 \rightarrow [0.7712, 0.773504]$

tag value lies b/w 0.7712 & 0.773504

when $x_n = 1$

$\therefore$ Fourth sequence = "1"

$0.772352 \longrightarrow$ "1 3 2 1"

"$a_1\ a_3\ a_2\ a_1$"

Algorithm for deciphering / Decoding the Tag

Step 1 : Initialize $l^{(0)} = 0$ & $u^{(0)} = 1$

2 : For each k find $t^* = \dfrac{(tag - l^{(K-1)})}{(u^{(K-1)} - l^{(K-1)})}$

3. Find the value of x_k for w/c $F_X(x_k - 1) \leq t^* \leq F_X(x_k)$

4. update u^k and l^k

5. Continue until the entire sequence has been decoded.

Tag generation with Scaling. $\Rightarrow .11000!$

or

Encode a sequence into binary value

$\left.\begin{matrix} 1 & 3 & 2 & 1 \\ a_1 & a_3 & a_2 & a_1 \end{matrix}\right\}$ binary fraction $[0,1]$

→ This Process Also known as incremental Encoding.

→ $[0, 1]$

lower half $[0.0, 0.5]$ — upper half $[0.5, 1.0]$

eq fn → send 0 — Send 1

→ Rescaling.

$E_1(x) = 2x$ — $E_2(x) = 2(x - 0.5)$

Ex. Encode the sequence $\underset{a_1\ a_3\ a_2\ a_1}{\text{"1 3 2 1"}}$ into binary code

$P(a_1) = 0.8$ $P(a_2) = 0.02$ $P(a_3) = 0.18$

Initially $l^0 = 0$ & $u^0 = 1$ $[0, 1)$

$l^n = l^{n-1} + (u^{n-1} - l^{n-1}) \times F_X(x_n - 1)$

$u^n = l^{n-1} + (u^{n-1} - l^{n-1}) \times F_X(x_n)$

For Ist Element "1"

$l^1 = l^0 + (u^0 - l^0) \times F_X(1-1) = 0 + (1-0) \times 0 = 0$

$u^1 = l^0 + (u^0 - l^0) \times F_X(1) = 0 + (1-0) \times 0.8 = 0.8$

Interval : $[0.0, 0.8)$ is not confied in any half.

So we Proceed further.

~~Now Rescale :-~~

$$\cancel{l^2 = 2(0.656 - 0.5) = 0.312}$$

$$\cancel{u^2 = 2(0.8 - 0.5) = 0.6}$$

~~[0.312, 0.6] Not confied in Ay half.~~

For Second Element "3".

$$l^2 = l^1 + (u^1 - l^1) \times F_X(3-1)$$
$$= 0 + 0.8 \times F_X(2) = 0.8 \times 0.82 = 0.656$$
$$u^2 = l^1 + (u^1 - l^1) \times F_X(3) = 0 + 0.8 \times 1 = 0.8$$

Interval [0.656, 0.8] confined in the upper half.

thus send "1"

Now perform scaling

$$E_2(x) = 2(x - 0.5)$$

$$l^2 = 2(0.656 - 0.5) = 0.312$$
$$u^2 = 2(0.8 - 0.5) = 0.6$$

[0.312, 0.6] Not confined so we proceed further.

~~Thus Inter~~

For 3rd Element "2"

$$l^3 = 0.312 + (0.6 - .312) \times F_X(2-1) = 0.5424$$
$$= 0.312 + (0.6 - 312) \times 0.8$$
$$u^3 = 0.312 + (0.6 - .312) \times F_X(2)$$
$$= 0.312 + (0.6 - .312) \times 0.82 = 0.54016$$

[0.5424, 0.54016] → upper half

send ⇒ '1'

Rescale

$$l^3 = 2(0.5424 - 0.5) = 0.0848$$
$$u^3 = 2(0.54816 - 0.5) = 0.09632$$

$[0.0848, -0.09632] \rightarrow$ lower half
$\downarrow$
Send "0"

Rescale

$$l^3 = 2 \times 0.0848 = 0.1696$$
$$u^3 = 2 \times 0.09632 = 0.19264$$

$[0.1696, 0.19264] \rightarrow$ lower half
$\downarrow$
Send '0"

Rescale.

$$l^3 = 2 \times 0.1696 = 0.3392$$
$$u^3 = 2 \times 0.19264 = 0.38528$$

$[0.3392, 0.38528) \rightarrow$ lower half $\Rightarrow$ Send "0"

Rescale-

$$l^3 = 2 \times .3392 = 0.6784$$
$$u^3 = 2 \times 0.38528 = 0.77056$$

$[0.6784, 0.77056] \rightarrow$ upper half. $\rightarrow$ Send "1"

Rescale

$$l^3 = 2 \times (0.6784 - 0.5) = 0.3568$$
$$u^4 = 2 \times (0.77056 - 0.5) = 0.54112$$

$[0.3568, 0.54112] \rightarrow$ Not confined any half.

Step 4

For 4^{th} element

$l^4 = 0.3568 + (0.54112 - 0.3568) \times Fx(0) = 0.3568$

$u^4 = 0.3568 + (0.54112 - 0.3568) \times Fx(1)$

$u^4 = 0.3568 * (0.54112 - 0.3568)$

$= 0.504256$

$[0.3568, 0.504256]$ X

Take binary value of Any of l^4 or u^4 let

let take binary of 0.5 → .1000 - - - -

Final Code

So Final Code → 1100 0100 - - -

TAG DEGENERATION WITH SCALING

[0, 1]

lower half [0, 0.5]   [0.5, 1] upper half

shift by 1 bit

Rescaling. $E_1 = 2x$ $E_2 = 2(x - 0.5)$

→ Differance b/w two smallest interval.

→ $2^{-k} <$ Difference

$\boxed{k = \quad}$

TAG DEGENERATION WITH SCALING

$$\begin{array}{ccc} & [0,1] & \\ \text{lower half } [0, 0.5] & & [0.5, 1] \text{ upper half} \\ & \text{shift by 1 bit} & \end{array}$$

Rescaling. $E_1 = 2x$ $\qquad E_2 = 2(x - 0.5)$

→ Difference b/w two smallest interval.

→ $2^{-k} <$ Difference

$$\boxed{k = \quad}$$

92, 116, 144, 154, 163,

178, 215, 229, 08, 131, 145,

Decode the sequence - "1100011000 .... 0"

$P(a_1) = 0.8$ $\quad$ $P(a_2) = 0.02$ $\quad$ $P(a_3) = 0.18$

$F_X(1) = 0.8$ $\quad$ $F_X(2) = 0.82$ $\quad$ $F_X(3) = 1.0$

→ In order to find out How many bits

$$[0.8, 0.82] = 0.$$

$$\text{Diff} = 0.82 - 0.8 = 0.02$$

$$2^{-k} < 0.02$$

let $k = 5$ $\rightarrow$ $\frac{1}{2^5} = \frac{1}{32} = 0.031$

let $k = 6$ $\rightarrow$ $\frac{1}{2^6} = 0.015 \leq 0.02$

So $\underline{K = 6}$

So Ist 6 bits are 110001 → 0.765625

$1\times 2^{-1} + 1\times 2^{-2} + 0\times 2^{-3} + 0\times 2^{-4} + 0 \times 2^{-5} + 1\times 2^{-6}$

$= 0.765625$

~~Now~~ $l^0 = 0$ $\quad$ $u^0 = 1$

$\underline{\text{Now}}$

$$l^1 = l^0 + (u^0 - l^0) \times F_X(x_n - 1)$$

$$= 0 + (1-0) \times F_X(x_n - 1) = F_X(x_n - 1)$$

$$u^1 = l^0 + (u^0 - l^0) \times F_X(x_n) = F_X(x_n)$$

let ~~$x_n = 1$~~ $\quad$ $[F_X(x_n - 1), F_X(x_n)]$

let $x_n = 1$ $\quad$ $[0, 0.8]$ $\quad$ Tag lies b/w Interval

So the Decoded Symbol → "1"

The Interval $[0, 0.8]$ not confined in any half.
$\therefore$ we move further

<u>for 2nd symbol -</u>

$$l^2 = l^1 + [u^1 - l^1] \times F_x(x_n - 1) = 0.8\, F_x(x_n - 1)$$

$$u^2 = l^1 + (u^1 - l^1) \times F_x(x_n) = 0.8\, F_x(x_n)$$

Interval: $[0.8\, F_x(x_n - 1),\ 0.8\, F_x(x_n)]$

let $x_n = 1$ $\quad [0, 0.64]$ ×

$\therefore$ $x_n = 2$ $\quad [0.64, 0.656]$ ×

$\therefore$ $x_n = 3$ $\quad [0.656, 0.8]$ ✓

So Decoded symbol is : "$\underline{\underline{3}}$"

The Interval $[0.656, 0.8]$ lies in upper half.
So we shift by 1 bit and Rescal.

<u>Rescale</u>

$$l^2 = 2(0.656 - 0.5) = 0.312$$

$$u^2 = 2(0.8 - 0.5) = 0.6$$

Next 6 bit → "100011" → $1 \times 2^{-1} + 0 \times 2^{-2} + 0 \times 2^{-3} + 0 \times 2^{-4} \times$
$= 0.546875$

$[0.312, 0.6]$ not confined in Any half. so

we calculate

For 3rd element

$l^3 = 0.312 + (0.288 \times Fx(x_n - 1))$

$u^3 = 0.312 + 0.288 \times Fx(x_n)$

$x_n = 1 \quad = \quad [0.312, 0.5424] \times$

$x_n = 1 \quad [0.5424, 0.54816]$ ✓ "2"

Now Tag. l's

Upper half. (shift by 1 bit & update]

$l^3 = 2 \times (0.5424 - 0.5) = 0.0848$

$u^3 = 2 \times (0.54816 - 0.5) = 0.09632$

Next 6 bit ⇒ 000110

Lower half (by 1 bit and update)

$l^3 = 2 \times 0.0848 = 0.1696$

$u^3 = 2 \times 0.$

Next 6 bit > 001100

Lowe

Binary Code for a Tag

→ If the mid-point of an interval is used as Tag $T_x(x)$, ~~$F_x(x)$~~ A binary code for $T_x(x)$ is the binary representation of number Truncated of $l(x) = \lceil \log(1/P(x)) \rceil + 1$ bit

→ Ex. $A = \{a_1, a_2, a_3, a_4\}$ with Probabilities $\{0.5, 0.25, 0.125, 0.125\}$, the binary code for each symbol is as follows

Sym	F_x	$\bar{T}_x$	In Binary	$\lceil \log \frac{1}{P(x)} \rceil + 1$	Code
1	0.5	0.25	.0100	2	01
2	0.75	0.625	.1010	3	101
3	0.875	0.8125	.1101	4	1101
4	1.0	0.9375	.1111	4	1111

Ex Sequence - 1

$l' = 0 + (1-0) F(0) = 0$

$u' = 0 + (1-0) F_x(1) = 0.5$

Interval → $\frac{0.5 + 0}{2} = 0.25$

Unique decodability of the code

→ Note that tag $T_X(X)$ uniquely specified the Interval $[F_X(x_{n-1}), F_X(x_n))$, if $\lfloor T_X(X) \rfloor_{l_X}$ is still in the interval, it is unique.

Since - $\lfloor T_X(X) \rfloor_{l_X} > F_X(x_{n-1})$ because

$$\frac{1}{2^{l_x}} < \frac{P(x)}{2} = T_X(x) - F_X(x-1),$$

we know $\lfloor T_X(X) \rfloor_{l_X}$ is in interval.

→ To show that the code is uniquely decodable, we can show that code is Prefix code.

This is true because $[\lfloor T_X(x) \rfloor_{l(x)} + \frac{1}{2^{l(x)}}) \subset [F_X(x-1), F_X(x)]$

Therefor, any other code out-side the interval will have different $l(x)$-bit prefix

Efficiency of code

Avg length of source $A^{(m)}$ is

$$l_{A^{(m)}} = \sum P(x) \cdot l(x)$$

$$= \sum P(x) \left[\left\lceil \log \frac{1}{P(x)} \right\rceil + 1 \right]$$

$$< \sum P(x) \left[\log \frac{1}{P(x)} + 1 + 1 \right]$$

$$= -\sum P(x) \log P(x) + 2 \sum P(x)$$

$$= H(X^m) + 2$$

Recall that for i.i.d. source $H(X^m) = mH(X)$

so $\boxed{H(X) \leq l_A \leq H(X) + \frac{2}{m}.}$

Comparison b/w Arithmatic Coding & Huffman Coding

(2) Average code length of m symbol sequence
→ Arithmatic code : $H(X) \leq l_A < H(X) + \frac{2}{m}$
→ Extended huffman : $H(X) \leq l_H < H(X) + 1/m$

→ Both code have same asymptotic Behaviour
→ Extended Huffman coding requires large CodeBook for m^n extended symbals while AC doesn't.

→ In general.
→ Small alphabet sets favours Huffman Coding
→ skewed distribution favours Arithmatic Cog

→ Arithmetic coding can adopt to input statistics easily.
→ Arithmetic codes allows us to code a sequence while in the typical Huffman code the all the symbals are encode.
→ It is easy to implement a system with multiple Arithmatic code.

→ It is much easiar to Adapt Arithmatic codes to changing I/p statistics.

→ (1) Arithmetic Code are not useful if we are coding the one symbal at a time.