EE5585 Data Compression February 5, 2013

Lecture 5

Instructor: Arya Mazumdar *Scribe: Kriti Bhargava*

Arithmetic Coding

To understand arithmetic coding better, there are some points that we need to keep in mind :

1. Shannon Fano Elias Coding: The basic idea of using a modified CDF rather than sorted probabilities to find the code word is extended in arithmetic coding.
2. X is a random variable $\in \mathbf{R}$
   $F_X(X)$ is uniform in the interval $[0, 1)$
3. For any $X \sim U[0, 1)$, binary expansion of X is a sequence of independent Bernoulli($\frac{1}{2}$) random variables.
4. The binary entropy function $h(p) = 1$ for p= $\frac{1}{2}$, that is, a sequence of independent Bernoulli($\frac{1}{2}$) random variables cannot be compressed any further. $X \to F_X(X)$ gives a uniform random variable and thus, making it incompressible.

Using an example : 0011011 with $p(0) = \frac{3}{4} = 1 - p \equiv q$; $p(1) = \frac{1}{4} = p$

Consider a continuous real random variable,

$$X = 0.x_1x_2x_3\ldots$$

It may not be independent, may be biased.

$$F_X(x) = P(X \leq x)$$

$$\begin{aligned} F_X(0011011) &= P(0.X_1X_2X_3\ldots \leq 0.0011011) \\ &= P(X_1 < 0) + P(X_1 = 0, X_2 < 0) \\ &\quad + P(X_1 = 0, X_2 = 0, X_3 < 1) + \cdots \\ &\quad + P(X_1 = 0, X_2 = 0, X_3 = 1, X_4 = 1, X_5 = 0, X_6 = 1, X_7 < 1) \\ &= q.0 + q^2.q + \cdots + q^3.p^3.q \end{aligned}$$

So, if we have probabilities of $k - 1$ bits, it can be extended further.

$$\sum_k P(x_1^{k-1}).x_k * q \text{ for } x_1^n = x_1x_2\ldots x_n$$

For finite length : $0.x_1x_2\ldots x_n$

$$[0.x_1x_2\ldots x_n00\ldots0; x_1x_2\ldots x_n111\ldots]$$

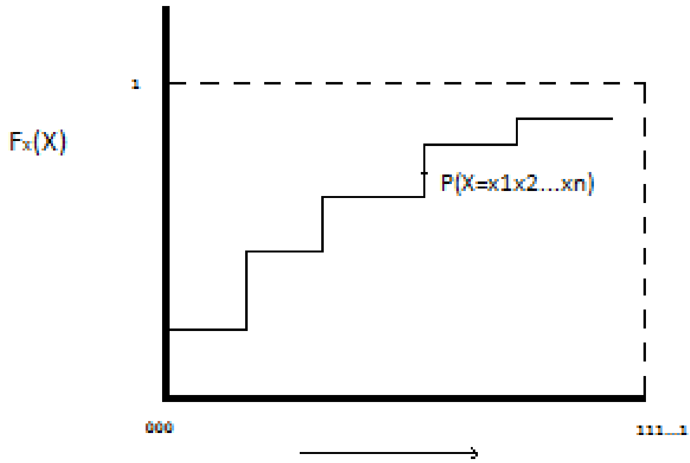
Size of this interval $= \frac{1}{2^n}$

$$F_X[\,] = [F_X(0.x_1x_2\ldots x_n00\ldots0), F_X(0.x_1x_2\ldots x_n111\ldots)]$$

All real numbers whose first n-bits are fixed, not a singleton set.

$$= P[X_1 = x_1, X_2 = x_2\ldots X_n = x_n]$$

Natural lexicographic order - 00,01,10,11



A seq of n random variable in SFE

$F_X^{-1}(u)$ = another real number whose first n bits are $x_1 x_2 \dots x_n$

Length (for compression) $= \lceil \log \frac{1}{p(x_1 \dots x_n)} \rceil + 1$

$$< H(x_1 \dots x_n) + 2$$

If each bit is independent, then for each bit

$$= \frac{1}{n}(H(x_1 \dots x_n) + 2)$$

For longer blocks, that is, as $n \to \infty$

$$\frac{H(x_1 \dots x_n)}{n} \to H(X); \frac{2}{n} \to 0$$

Application of arithmetic coding is mostly in fax machines and a few other JPEG standards.

Statistics (probability; the number of ones etc.) are not always available. If we know that source generates a sequence with a fixed number of ones but that number is unknown, a penalty is to be paid of sending the information of the whole sequence as property of the system is not known.
Thus, for a sequence like 001100110110, in general, information sent is:

$$\log_2 n + \log_2 \binom{n}{k}$$

Since $\log_2 \binom{n}{k} = \frac{c}{\sqrt{n}} 2^{nh(\frac{k}{n})}$

where $h(x) = -x \log x - (1-x) \log(1-x)$

$$= \frac{1}{n}[\log_2 n + nh(\frac{k}{n}) - \frac{1}{2} \log_2 n + \log_2 c]$$

$$= h(\tfrac{k}{n}) + \tfrac{1}{2n}\log_2 n + \tfrac{\log_2 c}{n}$$

where $\frac{\log_2 c}{n} \to 0$ and $\frac{1}{2n}\log_2 n$ is the penalty for not knowing enough information

$$0010011000\ldots$$
$$\chi = \{1, 2, 3, \ldots, m\}$$

We need an estimate of $p(0) = 1 - p$ and $p(1) = p$. For this, we need to go through the whole sequence and from the laws of large numbers, calculate the estimate of p and then move on to applying Huffman or Arithmetic coding. But this is not sequential.

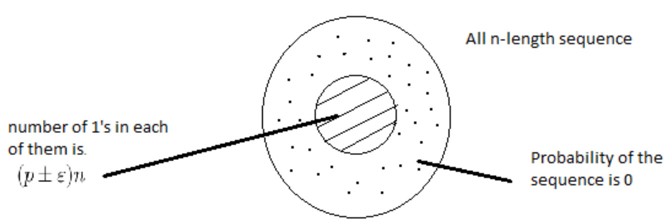
$P(\text{number of ones in the n length sequence} = w) = \binom{n}{w}p^w(1-p)^{n-w}$
{ Binomial distribution, therefore $\sum_{w=0}^{n}\binom{n}{w}p^w(1-p)^{n-w} = 1$}

$$\begin{aligned}\binom{n}{w}p^w(1-p)^{n-w} &= \frac{c}{\sqrt{n}}2^{nh(\frac{w}{n})}p^w(1-p)^{n-w}\\ &= \frac{c}{\sqrt{n}}2^{n(-\frac{w}{n}\log\frac{w}{n}-(1-\frac{w}{n})\log(1-\frac{w}{n}))}2^{w\log p}2^{(n-w)\log(1-p)}\\ &= \frac{c}{\sqrt{n}}2^{-n(\frac{w}{n}\log\frac{\frac{w}{n}}{p}+(1-\frac{w}{n})\log\frac{1-\frac{w}{n}}{1-p})}\\ &= \frac{c}{\sqrt{n}}2^{-nD(\frac{w}{n}\|p)}\end{aligned}$$

Suppose $\frac{w}{n} = \omega$.

$D(\omega \parallel p) \geq 0$
with eqaulity only if $\omega = p$
$\mid \omega - p \mid \geq \varepsilon > 0$
If $D(\omega \parallel p) = \text{positive}$; $2^{-n(+ve)} \to 0$
It is non-zero when ω is very close to p, i.e, $w \simeq pn$

Typical Set



Size of the typical set for small arbitrary ε say is $A_{n,p,\varepsilon} = \sum_{i=(p-\varepsilon)n}^{(p+\varepsilon)n}\binom{n}{i}$. Clearly

$$\binom{n}{pn} \leq A_{n,p,\varepsilon} \leq (2\varepsilon n + 1)\binom{n}{(p+\epsilon)n}$$

For very small ϵ, $A_{n,p,\varepsilon} \sim 2^{nh(p)}$
A file cannot be compressed beyond $h(p)$, compress up to $nh(p)$ bits

Rate of compression $\sim h(p)$ {Even for large alphabets}

For unknown p, the rate $= h(p)+$ error
As the block length is increased and made larger, there exists coding scheme for which error~ 0. One such coding scheme is Lempel Ziv coding scheme which is described next.

Lempel Ziv Coding

There are two kinds of Lempel Ziv Coding :

1. Sliding Window - Also known as LZ77.

2. Tree structured - First appeared in a paper in 1978. Also known as LZ78

Tree Structured Coding

Tree Structured is a dictionary based algorithm where a table is used for reference to determine the coding. In case of a tree structured algorithm, previous bits are referred to find out the sub sequences as we traverse along the whole sequence. This coding scheme is said to compress the sequence to the entropy of the sequence for large block lengths.

Take the following sequence - 0001101101110101011001011

For Tree Structured algorithm

- Parse the sequence : Traverse the whole sequence and parse the sequence into shorter sub sequences(phrases) where each sub sequence has not been visited earlier.

Numbering(Marker)	:1 2 3 4 5 6 7 8 9 10 11
Parsed Sequence	:0|00|1|10|11|01|110|101|011|001|011

- Encoding: For each sub sequence formed, encoding is done. A simple numbering system is used to mark the parsed string and the code is defined using marker,last bit in the sub sequence. This marker is called tuple. The final code word is made of the binary representation of the tuple and the last bit.
  For the above example, encoding is as follows :

  $(0,0)\ (1,0)\ (0,1)\ (3,0)\ (3,1)\ (1,1)\ (5,0)\ (4,1)\ (6,0)\ (2,1)\ldots$

$c(n)$ = number of passed sub sequences

For each sub sequence -
Number of bits required $= \log c(n) + 1$
Total $= c(n)(\log c(n) + 1)$

Rate of compression $= \frac{c(n)(\log c(n)+1)}{n}$