EE5585 Data Compression January 29, 2013

Lecture 3

Instructor: Arya Mazumdar *Scribe: Katie Moenkhaus*

Uniquely Decodable Codes

Recall that for a uniquely decodable code with source set $\mathcal{X}$, if $l(x)$ is the length of a codeword with $x \in \mathcal{X}$ then Kraft's Inequality is satisfied

$$\sum_{x\in\mathcal{X}} 2^{-l(x)} \leq 1 \tag{1}$$

A prefix code is a kind of uniquely decodable code in which no valid codeword is a prefix of any other codeword. These are generally easier to understand and construct.

Theorem 1 *For any uniquely decodable code C, let $L(C)$ be the average number of bits per symbol, $p(x)$ be the probability of the occurrence of symbol x, and $H(X)$ be the entropy of the source. Then*

$$L(C) = \sum_{x\in\mathcal{X}} l(x)p(x) \geq H(X) = -\sum_{x\in\mathcal{X}} p(x)\log_2(x) \tag{2}$$

Theorem 1 signifies that the entropy function puts a fundamental lower bound on the average number of bits per symbol.

Proof: Rearranging equation (2) leads to

$$\begin{aligned} L(C) - H(X) &\geq \sum_{x\in\mathcal{X}} l(x)p(x) + \sum_{x\in\mathcal{X}} p(x)\log_2(x) \\ &= -\sum_{x\in\mathcal{X}} p(x)\log_2 2^{-l(x)} + \sum_{x\in\mathcal{X}} p(x)\log_2 p(x) \\ &= -\sum_{x\in\mathcal{X}} p(x)\log_2\left(\frac{2^{-l(x)}}{\sum_{x\in\mathcal{X}} 2^{-l(x)}}\right) - \sum_{x\in\mathcal{X}} p(x)\log_2\left(\sum_{x\in\mathcal{X}} 2^{-l(x)}\right) + \sum_{x\in\mathcal{X}} p(x)\log_2(p(x)) \end{aligned}$$

Notice that

$$0 \leq \frac{2^{-l(x)}}{\sum_{x\in\mathcal{X}} 2^{-l(x)}} \equiv r(x) \leq 1 \tag{3}$$

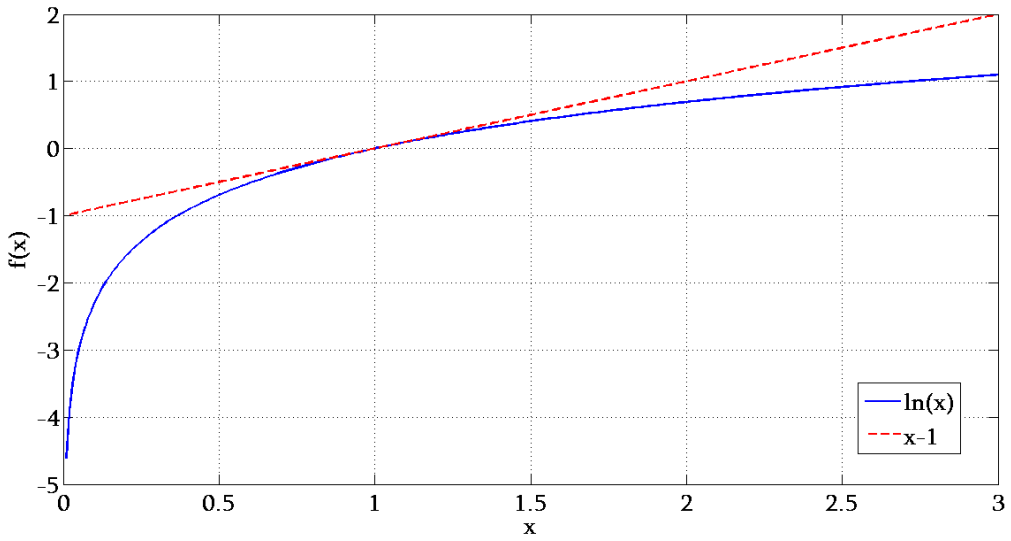
Combining the above equations leads to

$$-\sum_{x\in\mathcal{X}} p(x)\log_2\left(\frac{r(x)}{p(x)}\right) - \log_2\left(\sum_{x\in\mathcal{X}} 2^{-l(x)}\right)\sum_{x\in\mathcal{X}} p(x) \tag{4}$$

We have, $\sum_{x\in\mathcal{X}} p(x) = 1$. Additionally, change the base of the logarithm in the first term from 2 to e using the change of base formula for logs.

$$-\log_2(e)\sum_{x\in\mathcal{X}} p(x)\ln\left(\frac{r(x)}{p(x)}\right) - \log_2\left(\sum_{x\in\mathcal{X}} 2^{-l(x)}\right) \tag{5}$$

Recall the plot of $\ln(x)$:



The tangent line to the curve of $\ln(x)$ at $x = 1$ is $f(x) = x - 1$. It follows that

$$\ln(x) \leq x - 1 \tag{6}$$

Using equation (5) in conjunction with equation (6) leads to

$$-\log_2(e)\sum_{x\in\mathcal{X}} p(x)\ln(\frac{r(x)}{p(x)}) - \log_2(\sum_{x\in\mathcal{X}} 2^{-l(x)}) \geq -\log_2(e)\sum_{x\in\mathcal{X}} p(x)(1-\frac{r(x)}{p(x)}) - \log_2(\sum_{x\in\mathcal{X}} 2^{-l(x)}) \tag{7}$$

Consider the term $\sum_{x\in\mathcal{X}} p(x)(1-\frac{r(x)}{p(x)})$. Distributing, this becomes $\sum_{x\in\mathcal{X}} p(x) - r(x) = \sum_{x\in\mathcal{X}} p(x) - \sum_{x\in\mathcal{X}} r(x)$. As previously discussed, $\sum_{x\in\mathcal{X}} p(x) = 1$. Also, because of how r(x) is defined, $\sum_{x\in\mathcal{X}} r(x) = 1$. The subtraction of the two is then 0, so this term simplifies to 0. The right-hand side of equation (10) simplifies to

$$-\log_2 \sum_{x\in\mathcal{X}} 2^{-l(x)} \tag{8}$$

From Kraft's Inequality, this sum is always less than or equal to 1. Since $-\log_2(1) = 0$,

$$L(C) \geq H(X) \tag{9}$$

This holds for any uniquely decodable code.

Note: if $\log_e(x) \equiv \ln(x)$ is used, the unit that follows is nats instead of bits.

Optimal Codes

For a code to be optimal, the average length of the codewords is minimal. Recall that for Shannon Codes, $l(x)$ is given by

$$l(x) = \lceil \log_2(\frac{1}{p(x)}) \rceil \tag{10}$$

It can be shown that this code satisfies Kraft's Inequality. Also, for Shannon Codes,

$$H(X) \leq L(C) \leq H(X) + 1 \tag{11}$$

The Shannon code is not optimal in all cases.

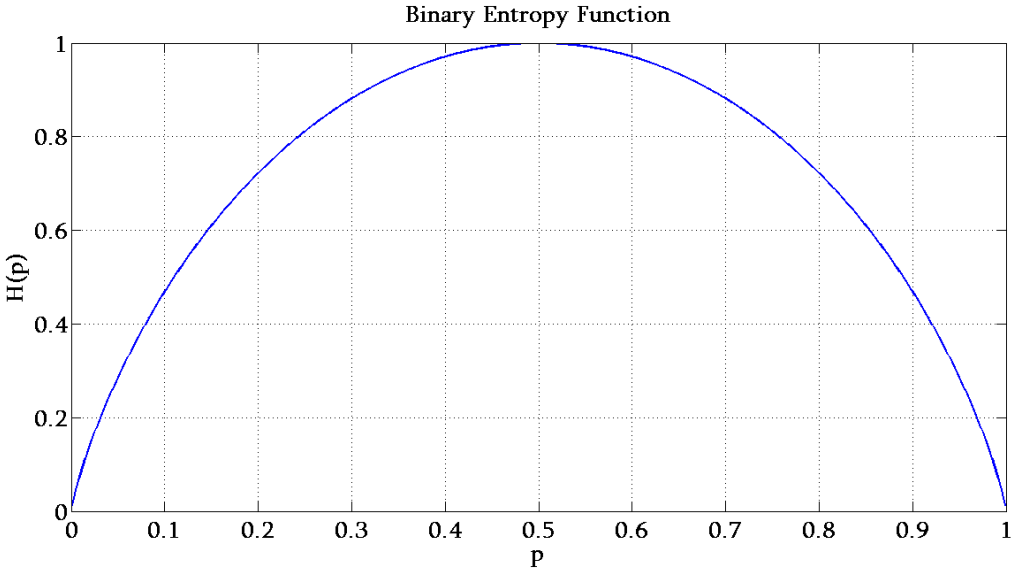
Example: Binary Entropy Function

$\mathcal{X} = \{0, 1\}$
$p(0) = p$
$p(1) = 1 - p$

The entropy of this set is

$$H(p) = -(p)\log_2(p) - (1-p)\log_2(1-p) \tag{12}$$

The graph of this function looks like



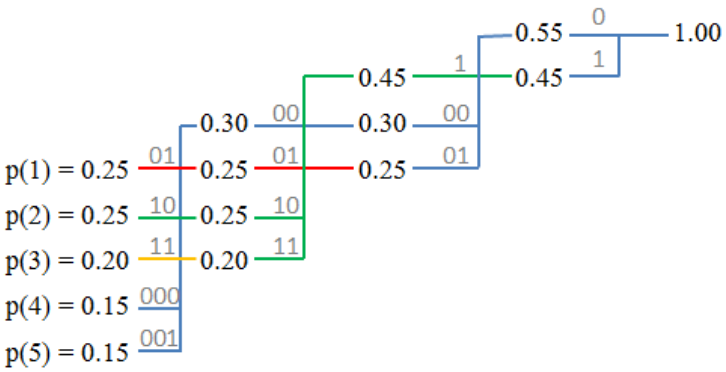
This function is symmetric about 1/2, i.e. $H(p) = H(1-p)$, and $H(p)$ achieves its maximum value at $p = 1/2$. This would represent data that contains equal numbers of zeros and ones. However, this is usually not the case for normal data, so the minimum number of bits needed to represent each symbol is actually less than one with a good compression algorithm.

Huffman Coding

Example: Suppose $\mathcal{X} = \{1, 2, 3, 4, 5\}$ with probabilities

x	$p(x)$
1	0.25
2	0.25
3	0.20
4	0.15
5	0.15

First, arrange the probabilities in decreasing order (already done in the above table). Next, add the probability of the two least likely symbols together to make a supersymbol with probability $p(4)+p(5)$. Repeat this process until only one supersymbol is left. This will be the number 1 and is the root of the tree. Binary values are then assigned starting at the root of the tree as follows.



Thus, the Huffman code for this set is

x	$C(x)$
1	01
2	10
3	11
4	000
5	001

The average number of bits per symbol is

$$L(C) = \sum_{x \in \mathcal{X}} p(x)l(x) = 0.25 * 2 + 0.25 * 2 + 0.2 * 2 + 0.15 * 3 + 0.15 * 3 = 2.3 \tag{13}$$

The entropy for this example, $H(X)$, is 2.2855. $L(C)$ being very close to $H(X)$ suggests that the Huffman Code is at least close to optimal. The tree method of assigning codewords presented above ensures that no codeword is a prefix of any other codeword, so the Huffman Code is a prefix code.

Optimality of the Huffman Code

A code is optimal when $\sum_{x \in \mathcal{X}} p(x)l(x) = L(C)$ is minimized. Suppose $\mathcal{X} = \{1, 2, ..., m\}$ with decreasing probabilities $p(1) \geq p(2) \geq ... \geq p(m)$.

Propteries of Optimal Codes:

- $l(1) \leq l(2) \leq ... \leq l(m)$
  Suppose an optimal code (C_{opt}) has $p(j) > p(k)$ but $l(j) > l(k)$. Define a new code in which the codewords $l(j)$ and $l(k)$ are switched.
  $L(C_{opt}) - L(C) = p(j)l(j) + p(k)l(k) - p(j)l(k) - p(k)l(j) = (l(j) - l(k))(p(j) - p(k)) > 0$
  This implies that C_{opt} cannot be the optimal code, and the optimal code has to have the mentioned ordering.

- $l(m-1) = l(m)$ i.e. the lengths of the longest two codewords are the same.
  Suppose, in an optimal code C, $C(m-1)$ and $C(m)$ have different lengths. Because of the prefix-free property, the last bit of the longer codeword could just be eliminated. The code will still be prefix free, and the resulting code will be better than the optimal code, a contradiction.
- $C(m-1)$ and $C(m)$, which are the codewords for the least likely symbols, differ only by the last bit. If they differ more than one bit, the last bit can be omitted from the codeword.

Codes that exhibit all of these properties are called canonical codes. Note: optimal codes are not necessarily unique; multiple optimal codes may exist for the same source.

In some situations, the Shannon code can be optimal. This happens when $\lceil \log_2(\frac{1}{p(x)}) \rceil = \log_2(\frac{1}{p(x)})$.

Example: $p(a) = 1/2$, $p(b) = 1/4$, $p(c) = 1/4$. For this probability distribution, the Shannon code is optimal because the probabilities are of the form $p(x) = 2^{-t}, t \in \mathbb{Z}$.

Theorem 2 *The Huffman code is optimal. For any other uniquely decodable code C and Huffman code C_H,*

$$L(C) \geq L(C_H) \tag{14}$$

Proof: For a source with two symbols, Huffman coding is optimal because each symbol is assigned one bit. An optimal canonical code, C^m_{opt}, for m symbols has codeword lengths $l(1) \leq l(2) \leq ... \leq l(m)$ and probabilities $p(1) \geq p(2) \geq ... \geq p(m)$. A new prefix code, C^{m-1}, is made from C^m_{opt} on $m-1$ symbols. The new set of probabilities is formed using $p(1), p(2), ..., p(m-2), p(m-1) + p(m)$. This is called Huffman reduction, in which the two least likely symbols are combined into a super symbol. The codewords for C^m_{opt} and C^{m-1} are the same until $m-2$, i.e. $C^m_{opt}(k) = C^{m-1}(k)$ for $k = \{1, 2, ..., m-2\}$. Then $C^{m-1}(m-1)$ is the codeword $C^m_{opt}(m)$ without the last bit. C^{m-1} is a prefix-free code, and

$$L(C^{m-1}) = L(C^m_{opt}) - p(m) - p(m-1) \tag{15}$$

Now let C^{m-1}_{opt} be an optimal code for the probability distribution $\{p(1), p(2), ..., p(m-2), p(m-1) + p(m)\}$. A new code, C^m, is formed from the probability distribution $\{p(1), p(2), ..., p(m-2), p(m-1), p(m)\}$. $C^m(m-1) = [C^{m-1}_{opt}(m-1), 0]$ and $C^m(m) = [C^{m-1}_{opt}(m-1), 1]$, where [ ] denotes append.

$$L(C^m) = L(C^{m-1}_{opt}) + p(m-1) + p(m) \tag{16}$$

Adding equations (18) and (19) leads to

$$L(C^{m-1}) + L(C^m) = L(C^m_{opt}) + L(C^{m-1}_{opt}) \tag{17}$$

This means

$$[L(C^m) - L(C^m_{opt})] + [L(C^{m-1}) - L(C^{m-1}_{opt})] = 0 \tag{18}$$

Each term in the above equation is nonnegative. They must individually sum to zero, meaning that $L(C^m) = L(C^m_{opt})$. Thus, C^m is an optimal code. The above procedure outlines one level of building a Huffman code. If the code is optimal at one level, the procedure can be extrapolated to an arbitrarily long Huffman code. Therefore, by induction, the Huffman code is optimal.