5. Coding for Efficiency

5.1 Introduction

Frequently it's necessary to transmit large quantities of data from one location or computer to another. If the message is very large, which is common in audio and video transmission, it's important to compress the data to save space and resource usage. It is this problem of compression that we examine here.

In performing the compression, we consider messages that are composed entirely of 1s and 0s as is the case in digital storage. Since we can assume nothing about the content of the message, compression must be performed using only information gleaned from these bits.

One way to compress the data is to divide the message into k-bit long blocks. In a message of length N, the number of total blocks will be N/k, or the smallest integer greater than or equal to N/k. The number of different k-bit patterns in the message is 2^k and each pattern will have a frequency associated with it for the message. For example, if k is 2, then the possible patterns are 00, 01, 10, and 11, and each one occurs a certain number of times in the message. From here out, we will refer to the frequency of each pattern as f(x) where x is the decimal representation of the binary pattern.

Is it possible to use this frequency information to shorten our message? Well, we can assign a "code word" to each message, and make the words that correspond to our frequently occurring pattern short, and those corresponding to rare patterns long. This raises several more questions: How much can we shorten our message? And how can we find an efficient algorithm for assigning code words for optimal compression?

The first question is answered by Shannon's Theorem, which defines a lower bound for the length of a message based on how much information is contained in it. Once we know this, then we can address the second question.

5.2 Shannon's Theorem

We know that if we shorten the message to M bits, then there are only 2^M possible messages we can create and distinguish between.

Since we know only the frequency of each block, there are numerous ways to order the blocks, and thus numerous possible messages. For example, if we know only that there are two A blocks and two B blocks, we can have AABB, BABA, ABAB, etc. Given a set of patterns and their corresponding frequencies in a set of n total blocks, the number of unique arrangements is equal to the number of orderings of n blocks, divided by the number of ways each specific arrangement can be made.

To rationalize this, consider arranging n unique blocks. Let's say we can write out R distinct reorderings of the blocks. Now imagine that the blocks are not all unique,

and thus there are actually Q appearances of each arrangement in our enumeration. That is, while the arrangements were formerly unique, now each arrangement will occur in this way Q of times. We can deduce then that the number of unique arrangements (A) is the number of possible reorderings (R) divided by the number of times that each can be made (Q).

The number of orderings of n objects is n! (obtained by simple permutations calculation) and since there are N/k blocks in our case, there are (N/k)! orderings. Divide this by the number of ways each specific arrangement can be created to get the number of unique arrangements. In our case, the blocks of one pattern x can be rearranged in a given message f(x)! times, as these are also simply permutations. Thus, the number of ways each arrangement can be made is $f(0)!f(1)!f(2)!\ldots f(2^k-1)!$.

Therefore, the number of unique arrangements is:

$$\frac{(N/k)!}{\prod_{i=0}^{2^k-1} f(i)!}$$

In other words, this is the number of messages that are N/k words long using 2^k different code words that we can create and will have to distinguish between. Since we know that M bits can be used to create 2^M different messages, the pigeonhole principle gives:

$$2^M > \frac{(N/k)!}{\prod_{i=0}^{2^k-1} f(i)!}$$

$$M > \lg \frac{(N/k)!}{\prod_{i=0}^{2^k-1} f(i)!}$$

This is our lower bound on the size of the message.

5.3 Entropy of Information

The concept of entropy is more commonly associated with the field of thermodynamics than information.

In physics, different parts of a system are frequently in different states. The entropy of a subsystem is a logarithmic measure of the diversity of the states of the system that can be observed in our small subsystem. These states generally appear with a frequency proportional to a factor of $e^{-E/kT}$ where E represents the energy of the state, and

T is the temperature of the system. The entropy is proportional to the logarithm of the sum over all states of the exponential factor just mentioned.

At high temperatures the exponential factor is close to 1 for many states and the entropy is high. At low temperature, only states with close to the lowest possible energy occur much at all, and the entropy is usually very low.

There are two laws of thermodynamics that involve entropy. The second law states that entropy of a system left on its own never decreases. That reflects the physical fact that states do not coalesce spontaneously, and a diverse system (having many states) will retain its diversity.

There is also a third law of thermodynamics which states that the entropy of a physical system approaches 0 near 0 temperature. This is of course only true of systems with very few low energy states. At low temperature only these states predominate in our subsystem and there is little diversity among the states. There is an exception to this law, and it is the substance known as water, but that is beyond our present horizon.

Shannon invented the concept of entropy of information, as a logarithmic measure of the diversity among potential messages that is wiped out upon receipt of a specific message. It thus provides a measure of how much information or how many bits you actually learn when you get a message.

In the case of information, if we know the frequency of our blocks, the diversity of potential messages comes entirely from the different orders in which these blocks can occur.

In our case, our frequency information tells us that the number of potential messages is as defined above. The log to the base 2 of this coefficient is the number of bits you must have in order to exceed the same number of potential messages, and is the number of bits you need supply in order to reduce the original diversity down to one particular message.

We can then define the entropy of information H to be:

$$\Sigma_{(0 \text{ to } N/k\ -1)}\ [-p(i)*\lg p(i)]$$

p(i) = k*f(i)/N = relative frequency of pattern i or fraction of blocks that have pattern i.

With this definition of entropy, the original bound on M can be massaged to form Shannon's Theorem:

If we compress a message of length N by dividing it into N/k blocks of length k each, using only frequency information contained in the frequencies f(s) of the various block patterns, we will need a number of binary digits for our message that is at least

*(N/k)*H*

and we can find a way to compress our message in this way that uses at most

(N/k)(H + 1) binary bits. (more on this in last section)

How exactly did we massage our original bound into Shannon' Theorem? By using Stirling's Formula as an approximation for n!

5.4 Stirling's Formula

To approximate n!, let's take a look at the power series of e^n.

$$e^n = 1 + n + \ldots n^{n-1}/(n-1)! + n^n/n! + n^{n+1}/(n+1)! + \ldots.$$

You can verify that the terms keep increasing in size up to the term $n^{n-1}/(n-1)!$, then the next term is the same size ($n^{n-1}/(n-1)! = n^n/n!$), and then the terms start decreasing, at ever increasing speeds, since the ratio of the kth term to the (k-1)st is n/k.

We can therefore estimate the left-hand-side, the exponential function, by the product of the largest term on the right and a measure of how many terms on the right make major contributions, in short by $n^n/n!$ *W where W is easily seen to be less than n.

Roughly, $W = (2\pi n + \pi/3)^{1/2} (1 + O(1/n))$ where the notation O(f(n)) means that as n goes to infinity its quantity behaves like f(n), in that its ratio to f(n) approaches a constant. The derivation of this is beyond the scope of these notes at the moment.

We can rearrange the approximation of e^n to give and approximation for n!

$$n! = (n/e)^n W$$

Using this approximation, we can alter the original pigeonhole bound on M to get Shannon's bound. Remember that f(x) = p(x)*N/k.

$$M > \lg \frac{(N/k)!}{f(0)!\, f(1)! \ldots f(2^k - 1)!}$$

$$M > \lg \frac{(N/k)!}{\left[\frac{N}{k} p(0)! \frac{N}{k} p(1)! \ldots \frac{N}{k} p(2^k - 1)!\right]}$$

$$M > \lg \frac{\left(\frac{N/k}{e}\right)^{N/k}(W_1)}{\left[\left(\frac{(N/k)p(0)}{e}\right)^{(N/k)p(0)} W_2 * \left(\frac{(N/k)p(1)}{e}\right)^{(N/k)p(1)} W_3 * \ldots\right]}$$

All the $\left(\frac{N/k}{e}\right)$ terms cancel out.

$M > \lg W_1 * [p(0)^{-N/k*p(0)} * p(1)^{-N/k*p(1)} \ldots] / (W2*W3\ldots)$

The lg W terms are not significant for large n.

$M > N/k * [-p(0)lgp(0) - p(1)lgp(1)\ldots -p(2^k-1)lg(2^k-1)]$

$M > N/k * H$ using our definition for H.

Thus, using Stirling's Formula and the concept of entropy of information, we have this bound on the number of bits needed in our compressed message.

5.5 Making a code obeying the Shannon's bound

Given that the frequency of each block q is f(q) and the length is l(q), the length of the entire m1essage will be:

$\Sigma_{\text{all q}} f(q)l(q)$

In forming a code, let's consider the very specific case, that each p(q) is the reciprocal of some power of 2. Recall that p(q) is the relative frequency of q. Since the sum of all p(q) must be 1, then the rarest two blocks (corresponding to the smallest p(q)) must be equal. Rationalize this fact by considering what the possible p(q) values are: ½, ¼, $^1/_8$ … In order for that sequence to sum to 1, there must be a point at which the sequence is stopped and the last frequency is duplicated. Try out some more possibilities to convince yourself.

Now, we take those two rare blocks and decide that we will choose a common code word for them, except that the last bit in one will be 0, and the last bit in the other will be 1. From now on, we will treat the two blocks as one block with double the frequency.

It follows that we can now assign a code word of length l(q) to each block of frequency p(q), such that l(q) = -lgp(q). For the two rare blocks, since they are treated as one block, they will be assigned a code word of length –lg[2*p(n)] where p(n) the frequency of each individual block. Then, to distinguish the two blocks, one of them has a 1 appended, and one has a 0 appended. Incidentally, -lg[2*p(q)] is exactly one less than

–lgp(q). With the final bit added, the two rare blocks have code words that are –lgp(q) in length, like all the other blocks.

With a code like this, the length of the total message will be:

-f(0)lgp(0) -f(1)lgp(1) -f(2)lgp(2)… which is what the Shannon bound gives. If the block frequencies are not inverse powers of 2, we can make them as such by lowering the frequencies by an appropriate amount. In the worst case, a frequency will have to be halved, in which case all of the lgp(q) terms in the length of the message will be 1 greater than they were originally. This is where the constant of 1 in the second half of Shannon's Theorem comes from.

So we can actually find a reasonably good code by rounding our relative frequencies down to inverse powers of 2 and assigning code words to have length given by their -lgp(q) values

But this code is not optimal, and we can do better with an easier procedure, which we will describe in the next section.

Exercises

Exercise 1 Consider a message having only two block types: 0 and 1. Given that the relative frequencies are p(0) and p(1), plot the entropy of the message over the range $0 < p(0) < 1$.

Exercise 2 Given the following block frequencies f(q), compute p(q) and the total entropy of the message: 1, 22, 11, 15, 2, 1, 2, 3, 45, 95, 2, 2, 1, 2, 25, 1

Exercise 3 Explain why a set of relative frequencies p(q) that are all negative powers of 2 must contain a duplicate frequency. Why must that frequency be the smallest in the set?

- Steven Kannan