EE5585 Data Compression February 12, 2013

Lecture 8

Instructor: Arya Mazumdar *Scribe: Viola Z. Dsouza*

Review - An intuitive look at lossless compression techniques studied so far

When we look at Huffman coding, the procedure looks intuitive; we combine the least probable symbols and proceed iteratively. It is not difficult to understand why this algorithm works. But in the case of Lempel-Ziv coding, although we've been told it is optimal, it is not immedaitely clear as to why so. To get an intuitive understanding of why these compression algorithms work, we need to grasp some basic underlying concepts, which will be further elaborated in this text.

In real life, we may deal with many deterministic as well as random events. For example, a question such as,"Will the Sun rise tomorrow?", leads to a deterministic answer, namely, "Yes". Another event would be to determine the outcome of tossing a coin. If we knew all the factors that come in to play in this event, such as the wind velocity, force, torques and the likes, we would classify this event as being deterministic. But as it turns out, in real life, these are too many factors to account for, and as such, this event is random.

These concepts are directly related to our job of designing an efficient compression algorithm. We want to design an algorithm which is universal, i.e., one which does not depend on the file to be encoded. The entire file to be encoded is taken to be one random variable.

What is information?

Consider an event of tossing a *biased* coin, where heads is more likely to appear than tails. Now if we observe heads as the outcome, it does not convey much information since we already knew that it was more probable to appear. On the other hand, if a tails were to have appeared, we would've gained more information, since such an occurence is rare. This example serves to show that the information conveyed is directly related to the probability of occurence of an event.

Let us define a function $f(p)$ on a random variable which conveys the information contained in it and depends on the probability of occurnce of that random variable. This function $f(p)$ satisfies the following 3 properties :

1. $f(p) \geq 0$;
2. $f(p)$ is a decreasing function of p;
3. $f(p_1, p_2) = f(p_1) + f(p_2)$;

A known function which satisfies all the above 3 properties is the *log* function. Thus, we can define $f(p)$ as :

$$f(p) = c \log_2 \frac{1}{p}$$

Thus, any function of the form $f(p) = \log_2 \frac{1}{p}$ (ignoring the constant c) will represent the information contained in the random variable. The average information contained is given by the expectation of f, i.e., $E[f(p)]$

Now if $Pr[E_1] = p_1$ and $Pr[E_2] = p_2$, where E_1 and E_2 are two events, then the average information contained in both of these events is calculated by taking the weighted average of information contained

in individual events i.e.,

$$Average\ information = p_1 \log_2 \frac{1}{p_1} + p_2 \log_2 \frac{1}{p_2}$$

This equation is nothing but the equation for calculating entropy, $H(p)$. Thus, the information obtained from the file is the entropy. While designing compression algorithms, the entropy sets the lower limit on the average number of bits per symbol which can be achieved without loss of information

Lempel-Ziv Coding

Since Lempel-Ziv compression is designed to operate when statistics of the files to be encoded are not known beforehand, it appears as though it is a random algorithm. But it still works exceptionally well, at least for independent symbols or stationary/Ergodic random processes. Let us examine one of the Lemmas which we've used before, namely

$$C(n) \leq \frac{n}{\log_2 n}$$

Where $C(n)$ represents the distinct parsed phrases and n is the total number of bits in the file.

Now this formula is a general formula for any parsing which results in distinct phrases, even if these phrases are not Lempel-ziv specific. The phrases which we get by the Lempel-Ziv algorithm have a certain relationship between each other, i.e they are not parsed randomly. Whereas the result given by the lemma does not take into account any such relation between the parsed phrases.

Another relation for distinct phrases is :

$$H(File) \geq C(n) \log_2 C(n)$$

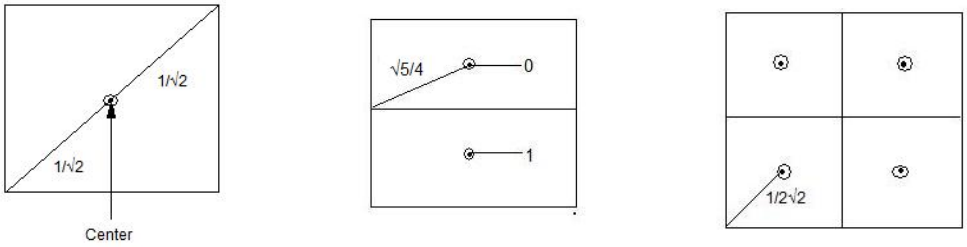
This is also a generic equation which does not take into account any specifications for Lempel-Ziv coding. Any algorithm which satisfies this equation will be optimal. Lempel-Ziv gives a way to accomplish this since the number of bits needed to be used is:

$$C(n)[\log_2(C(n)) + 1]$$

Quantization (Lossy Coding)

In lossy encoding, by giving up some information, we are able to compress the file in such a way that the average number of bits used to represent a symbol after encoding may be reduced to a value lower than the entropy $H(.)$. For example, when we implement lossy encoding for images, we discard some of the redundant pixel values. While reconstructing, these pixels get corrupted because they cannot be successfully recovered. But the human eye is not able to discern such a minute change and hence discarding such redundant information doesn't have much effect on the quality of the image. We can achieve a high rate of compression and save some bandwidth as well.

Now consider a square of area 1 $unit^2$. Each point in the square represents a file, i.e. the file is a coordinate in this square. If we were to perform quantization of any given file such that it can be represented with a certain number of bits, we would have to choose centers in this square to which we can map the files. Let us examine a few cases based on the number of bits used to represent the coded file.



(a) 0 bitl (b) 1 bit (c) 2 bits

Figure 1: Compression to different number of bits

1. Compression to 0 bits

   In this type of representation, we choose 1 point as the center to which all points in the square would be mapped to. Therefore, irrespective of the input, the code will always be same. Figure 1(a) represents such a scheme.

   The worst-case distortion is defined as the Euclidean distance between the center and the farthest point in the square. Here, this distance is given by $\frac{1}{\sqrt{2}}$.

2. Compression to 1 bit

   One way to choose centers for mapping files is as shown in figure 1(b). Here, the maximum distortion is $\frac{\sqrt{5}}{4}$. Comparing with case 1, we can see that the worst-case distortion has decreased, but we do not know if the centers chosen are optimal or not.

3. Compression to 2 bits

   A possible selection of centers is shown in figure 1(c). Here, worst-case distortion is given by $\frac{1}{2\sqrt{2}}$.

So far we have measured distortion as a function of the Euclidean distance. We will now move towards a different measure of distortion known as *Hamming distance*.

Definition : Hamming distance between two files of equal length is the number of bit positions in which the files are different.

Consider two files X_1^n and Y_1^n given by :

$$X_1^n = X_1 X_2 \dots X_n \in \{0,1\}$$

$$Y_1^n = Y_1 Y_2 \dots Y_n \in \{0,1\}$$

The *Hamming distance* between X_1^n and Y_1^n is given by

$$d(X_1^n, Y_1^n) = \text{ Number of disagreements}$$

Hamming distance will always satisfy the following 3 properties :

1. $d(x,y) \geq 0$ with equality if and only if $x = y$
2. $d(x,y) = d(y,x)$
3. $d(x,z) \leq d(x,y) + d(y,z)\ \forall y$

Covering Problem

Consider a region enclosed by the black boundary as shown in figure 2. Our first task would be to find 'Centers' for the given region. Once we have found these, we will map our given point (file) to the closest center (in terms of Hamming distance) .

Suppose we have chosen centers as shown in figure 2 (Red Dots). Let the worst case distance (distance of the farthest point in the region from the center) be d_W. We will now construct spheres in our given region centered at the red dots with radius d_W. Union of all such spheres must cover all the points in the space. Set of all points located in the sphere is given by :

$$\{y : d(x, y) \leq d_W\}$$

The *volume* of the sphere is given by the size of set of all points within the sphere.

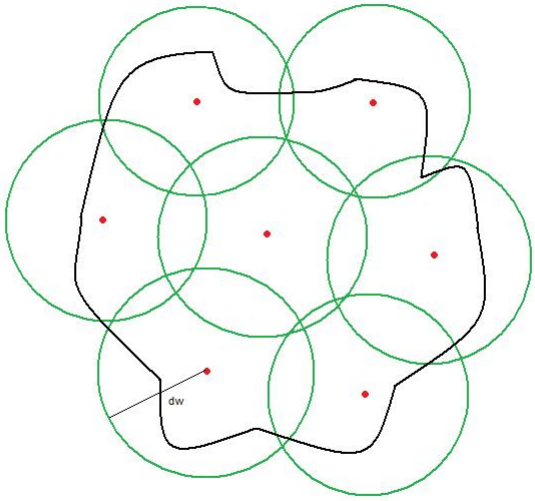


Figure 2: Example of optimal covering

If the total number of centers is given by M and the volume of each sphere is given by S, then

$$SM \geq U \tag{1}$$

where U is the volume of the entire space.

To understand the use of this inequality, we go back to the example of figure 1(c), where we used 2 bits to represent the coded output. Suppose we want to achieve a worst-case distortion $r = \frac{\sqrt{2}}{4}$, then using equation (1), we get

$$\begin{aligned} \pi r^2 \times M &\geq 1 \\ \pi \times \frac{2}{16} \times M &\geq 1 \\ M &\geq \frac{8}{\pi} \\ M &\geq 3 \end{aligned}$$

This shows that to achieve a distortion of $\frac{\sqrt{2}}{4}$, we need at least 3 points.

Example:
Consider a file source of 3 bits. To encode this particular set of files, we first map each file to a code that differs only in 1 bit, i.e. is a Hamming distance 1 away from the file. Now, we are left with only 4 values. We can now assign a 2 bit codeword to represent these.

$$\begin{matrix} 001 \\ 000 \end{matrix} \rangle\ 000 \text{ — } 00$$

$$\begin{matrix} 011 \\ 010 \end{matrix} \rangle\ 010 \text{ — } 01$$

$$\begin{matrix} 001 \\ 000 \end{matrix} \rangle\ 000 \text{ — } 00$$

$$\begin{matrix} 001 \\ 000 \end{matrix} \rangle\ 000 \text{ — } 00$$

Using the codewords that we have assigned, the worst-case distortion measured in terms of the Hamming distance is 1bit (absolute distortion). The normalized distortion is given by $\frac{1}{3}$ and the compression ratio achieved is $\frac{2}{3}$. If we were to calculate distortion based on probability, the *average distortion* would be $\frac{1}{2}$.

Now let us try to calculate the minimum number of centers we would require for the file space provided if we want a maximum distortion of 1 bit. Consider the file 000. Here, the volume of each sphere will be the number of points, which are at a distance less than or equal to 1 from 000. These points would be :

$$010$$

$$100$$

$$001$$

$$000$$

The total number of such points is 4. Therefore, the volume of the sphere is 4. Substituting our values in equation (1) we get :

$$\begin{aligned} M \times 4 &\geq 8 \\ M &\geq 2 \end{aligned}$$

Thus, we require minimum two centers to achieve a maximum distortion of 1 bit. This goes to show that the centers chosen for this example are not optimal.

Let us reexamine equation (1). By rearranging the terms we obtain :

$$M \geq \frac{U}{S} \tag{2}$$

Taking $\log_2$ on both sides we get

$$\log_2 M \geq \log_2 \frac{U}{S}$$

Here, $\log_2 M$ represents the number of bits required to represent the code. In practice, $\log_2 \frac{U}{S}$ is a good estimate for the lower bound for very large files.

Volume of sphere in Hamming Space

We have previously defined the volume of a sphere of radius r to be the number of points in space which are at a distance less than or equal to r from the center. Irrespective of the center location, the number of points within a distance r from the center are always going to be the same. Suppose we have an all zero file of n bits.

$$0000000\ldots0$$

Number of points at a distance 1 from this file = n
Number of points at a distance 2 from this file = $\binom{n}{2}$
Number of points at a distance 3 from this file = $\binom{n}{3}$
$\vdots$
Number of points at a distance n from this file = $\binom{n}{n}$
Thus, the volume of a sphere with radius r and center as the all zero n-bit file is:

$$\text{Volume } = \sum_{i=0}^{r} \binom{n}{i}$$

Now, the volume of the entire space, i.e U is given by:

$$U = 2^n$$

To achieve a worst case distortion of r, the number of centers required calculated using equation (2) is :

$$M \geq \frac{2^n}{\sum_{i=0}^{r} \binom{n}{i}}$$

Taking $\log_2$ on both sides, we obtain the number of bits required for representing M centers, given by:

$$\log_2 M \geq n - \log_2 \sum_{i=0}^{r} \binom{n}{i}$$

Thus the rate of compression, R is given by :

$$R \geq 1 - \frac{1}{n} \log_2 \sum_{i=0}^{r} \binom{n}{i} \tag{3}$$

Now as compression code designers, we would like to ensure a high rate of compression alongwith minimum distortion, i.e we want to decrease both R and r. But looking at equation (3), it is quite clear that both R and r cannot decrease at the same time. To observe this relationship clearly, lets take a look at the plot of compression rate v/s the worst case distortion distance in figure 3. To ensure that equation (3) holds, we want to operate in the shaded region shown in figure 3.

Let us now move on from absolute distortion measures to a normalized one. The normalized distortion D given by :

$$D = \frac{r}{n}$$

i.e.,

$$r = Dn$$

where $0 \leq D \leq 1$

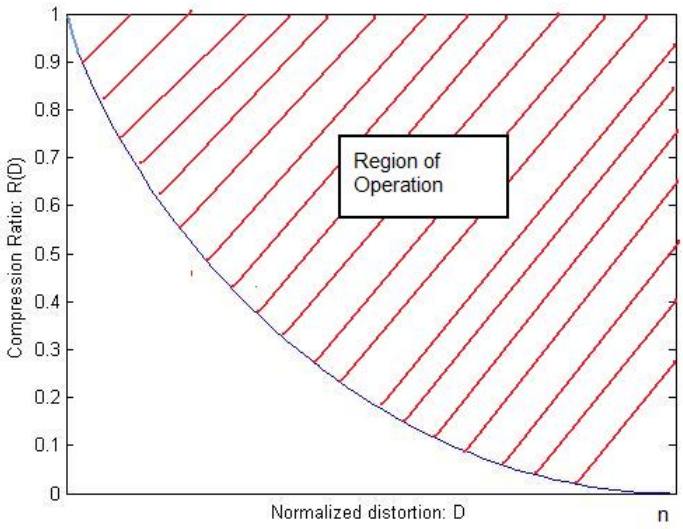


Figure 3: Plot of number of bits(n) v/s compression ratio(R)

Let $R(D)$ represent the optimal compression rate given D is the normalized distortion and number of bits, $n \to \infty$. Substituting in equation (3) we get:

$$R(D) \geq 1 - \lim_{n\to\infty} \log_2 \sum_{i=0}^{r} \binom{n}{i} \tag{4}$$

But recall that

$$\binom{n}{i} \approx \frac{c}{\sqrt{n}} 2^{nH(\frac{k}{n})}$$

where

$$H(x) = -x \log_2 x - (1-x)\log_2(1-x)$$

On using Stirling's inequality and performing necessary calcukations, the result obtained is

$$\frac{2^{nH(D)}}{\sqrt{8D(1-D)n}} \leq \binom{n}{Dn} \leq \frac{2^{nH(D)}}{\sqrt{2\pi D(1-D)n}}$$

i.e.,

$$\frac{2^{nH(D)}}{\sqrt{8D(1-D)n}} \leq \sum_{i=0}^{Dn} \binom{n}{i} \leq 2^{nH(D)}$$

This approximation holds for any $D \leq \frac{1}{2}$. Using this result in equation (4) we get

$$R(D) \geq 1 - H(D)$$

This curve of compression ratio $R(D)$ v/s normalized compression ratio D is plotted in figure 4. If we allow normalized distortion to be $\frac{1}{2}$, we can have a code of zero rate. Distortion of $\frac{1}{2}$ means that given a file of length n, the codeword developed is at most a distance $\frac{n}{2}$ from the file, i.e distance between estimate and actual file is at most $\frac{n}{2}$. If size of file is 2, we will require only 1 bit to write it.

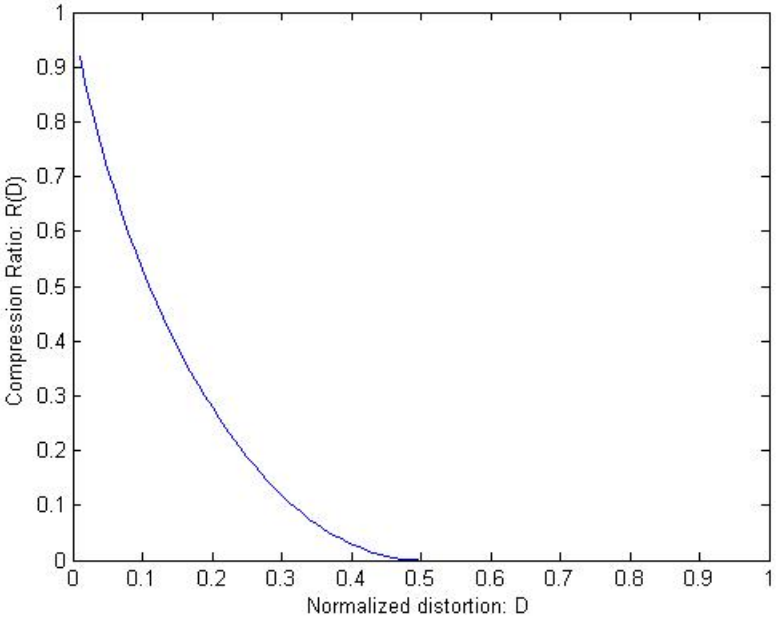


Figure 4: Plot of normalized compression ratio D v/s compression ratio R(D)