Lempel Ziv 1978

January 21, 2013

1 Motivation

An encoder E is said to be *information lossless* if for all initial states z and all strings x, the triple $(z,\ \text{output}(z,x),\ \text{state}(z,x))$ uniquely determines x.

E is said to be *information lossless of finite order* if for all initial states z and any string x, the pair $(z,\ \text{output}(z,x))$ uniquely determines x. In particular, the final state need not be transmitted to decode x. One way to ensure this is by forbidding λ-transitions.

A coding theorem, Theorem **??** is proved using an information lossless of finite-order constructionm while its converse **??** is proved under the information lossless assumption.

The compression ratio achieved by encoder E on string $x \in A^n$ is

$$\rho(E, x[1 \ldots n]) = \frac{L(y[1 \ldots n])}{n \log_2 \alpha}.$$

The minimum of $\rho(E, x[1 \ldots n])$ over the class $E(s)$ of all finite-state information lossless encoders with input alphabet of size α and at most s number of states is denoted by $\rho(E(s), x[1 \ldots n])$. This quantity depends on s, x and n. (Please note that $E(s)$ depends on x. The optimal compressor for another string might be different.)

Further, let

$$\rho(E(s), x) \quad = \quad \limsup_{n \to \infty} \rho(E(s), x[1 \ldots n]).$$

This quantity is the asymptotic compressibility of x using finite-state compressors using s states.

Now, as the number of states increases, we get better compression - to see this, observe that every s-state compressor can be trivially made into a compressor having $s+1$ states by adding an unreachable state. Hence the compressibility achieved by compressors with $s+1$ states is at most as high as that achieved by those with only s states. Hence, we can define

$$\rho(x) = \lim_{s \to \infty} \rho(E(s), x).$$

The latter quantity that depends only on x is referred to as the finite-state compressibility of the infinite sequence x. This quantity, for every sequence x, lies in the closed unit interval.

1.1 Results about individual sequences

We derive a lower bound for $\rho(x)$ for every sequence x, using a parsing argument. In this part, we will argue that every *information-lossless compressor of finite order* will have to output strings of a particular length. This is called the **converse-to-coding theorem**.

We derive an upper bound for $\rho(x)$ by constructing the disjoint-block Lempel-Ziv compressor, whose s-sections (we will define this) are provably almost optimal over all finite-state *information lossless compressors* having at most s states. This is the constructive **coding theorem**.

Even over a single infinite sequence, we can take a probabilistic viewpoint by considering the empirical distribution on n-long strings defined by the particular infinite sequence. In this case, we prove that the entropy of the distribution of strings over A^* is precisely the finite-state compressibility of the sequence x, for every sequence x.

1.2 Probabilistic viewpoint

In addition, we can consider a probabilistic viewpoint, that the infinite sequence x is drawn from an ergodic source with entropy H (the entropy being defined on the source itself), or a stationary source with entropy H.

If the source is ergodic, we have a result that *with probability 1* (but not necessarily for every sequence x, that the entropy of the source matches the finite-state compressibility of x. [1]

If the source is stationary but not necessarily ergodic, then we have a result in L_1 convergence - that the expected value of the compressibility over all the sequences, is the entropy of the source.

2 Coding Theorem - Algorithm

We will give an algorithm $\mathcal{E}$ to compress sequence and prove its performance characteristics.

Theorem 2.1. *For every $n > 0$ there is an ILF encoder $\mathcal{E}$ with $s(n)$ states that implements a block to variable code with the following properties.*

In what follows, for a finite string x, $c(x)$ is the maximum number of distinct *phrases whose concatenation forms x.*

1. *$\forall n, i$ and every input block $x[1 \dots n]$, the compression ratio of $\mathcal{E}$ satisfies*

$$\rho(\mathcal{E}, x[in+1 \dots (i+1)n]) \leq \frac{c(x[in+1 \dots (i+1)n])}{n \log \alpha} \log[2\alpha(c(x[(in+1 \dots (i+1)n] \ + \ 1))].$$

2. *For every finite s,*

$$\rho(\mathcal{E}, x[1 \dots n]) \leq \rho(E(s), x[1 \dots n]) + \delta_s(n),$$

   with

$$\lim_{n \to \infty} \delta_s(n) = 0.$$

3. *Given an infinite input sequence ω, for any $\epsilon > 0$,*

$$\rho(\mathcal{E}, \omega[1 \dots n]) \leq \rho(\omega) + \delta_\epsilon(\omega, n),$$

   where

$$\lim_{n \to \infty} \delta_\epsilon(\omega, n) = \epsilon.$$

We will give the proof in two parts. First we give the algorithm - the encoder and the decoder. Then, we will show that it has the above properties.

[1] Shields [?] strengthened this result later to show that for every sequence x, the same result holds.

2.1 Algorithm

§1. For the encoder $\mathcal{E}$, we have an ILF finite-state machine that implementsa conctenated coing scheme by combining (a) a fixed block-to-variable block outer code with (b) a state-dependent variable-to-variable inner code.

The inner code encodes sequentially, growing segments of an n-long input block, where n is fairly large. After the encoder finishes scanning an n-block, it returns to the initial state, thus "forgetting" its past.

In an adaptive dictionary scheme, we need to build a dictionary of phrases, and have a parsing scheme to identify the next phrase in the input string. The segments of a block which form the "dictionary input" of the inner coder will be constructed using an *incremental parsing procedure*. The procedure is greedy and sequential.

Suppose at some stage in the algorithm, we have segmented $x[1 \dots n]$ into distinct phrases. The new phrase is the shortest prefix of the unparsed part, that is, $x[n \dots]$ which is different from all current phrases in the dictionary.

Let the current parsing be

$$x[1 \dots n] = x[1 \dots n_1]x[n_1 + 1 \dots n_2] \dots x[n_p + 1 \dots n].$$

This parsing is called incremental, if the first p phrases are all distinct, and if for all $j = 1, 2, \dots p+1$, there is a positive integer $i < j$ such that $x[n_{i-1}+1 \dots n_i] = x[n_{j-1}+1 \dots n_j]$ - that is, there is some earlier phrase at position i such that the phrase at position j is exactly the phrase at position i extended by 1 bit. Since we need this operation frequently, let us define this as a function. The function $d : \Sigma^* \to \Sigma^*$ is defined by $d(w) = w[1 \dots |w| - 1]$, that is, the longest proper prefix of w.

Thus for every $j = 1, 2, \dots, p+1$, there is a unique nonnegative integer $\pi(j) = i$, where $i < j$, such that $d(x[n_{j-1} + 1 \dots n_j]) = x[n_{i-1} + 1 \dots n_i]$.

The parsing rule is as follows. To determine the j^{th} phrase, $j = 1, 2, \dots, p+1$, let n_j be the largest integer less than n such that $d(x[n_{j-1} + 1 \dots n_j])$ is among the phrases $1, 2, \dots, j-1$.

The phrase $x[n_{j-1} + 1 \dots n_j]$ is encoded in binary as

$$I(x[n_{j-1} + 1 \dots n_j]) = \pi(j)\alpha \ + \ I_A(x[n_j]),$$

where $I_A : A \to \{0, 1\}^*$ is a binary encoding of each of the symbols in A.

It is easy to see that

$$0 \le I(x[n_{j-1} + 1 \dots n_j]) \le (j-1)\alpha + \alpha - 1 = j\alpha - 1.$$

Thus the length of the encoding of the j^{th} phrase is $\lceil \log(j\alpha) \rceil$.

§2. **Decoder.** The decoder gets a binary string. It has access to the alphabet encoding function I_α. It has to output the original message, a string in A^*. (Alternatively, we can see the decoder as transforming an infinite binary sequence to an infinite sequence in A^∞.) It proceeds inductively as follows.

We use three variables, j, which is the index of the next phrase, k_j, the , and n_j, the length of the j^{th} phrase. At the beginning, set $j = 0$, $k_j = 0$ and $n_j = 0$.

In addition, we keep a dictionary, which is a variable-length-array of variable-length phrases.

Suppose we have decoded up to phrase $j - 1$, and the values k_j and n_j reflect this.

Increment k_j by $\lceil \log(j+1)\alpha \rceil$. Take the next k_j bits from the input binary sequence. Let $I(x[n_j + 1 \dots n_{j+1}])$ be the integer whose representation is given by these bits. Determine the unique nonnegative integers i and r which satisfy

$$I(x[n_j + 1 \dots n_{j+1}]) = i\alpha \ + \ r, \qquad 0 \le r \le \alpha - 1.$$

Now, we can extract the last letter of the $j+1^{\text{st}}$ phrase from r.

Similarly, the prefix of the $(j+1)^{\text{st}}$ phrase is the i^{th} entry in the dictionary. Insert $x[n_j+1\ldots n_{j+1}]$ as the $(j+1)^{\text{st}}$ entry in the dictionary.

If $n_j + \ (n_i - n_{i-1}) + 1 \geq n$, then we are done. Otherwise, set $n_{j+1} = n_j + \ (n_i - n_{i-1}) + 1$, increment j by 1, and recurse. (If the decoder is decoding an infinite binary sequence, then it does not have a stopping condition.)

2.2 Properties

§1. First, some easy observations about the parsing in the last section. The first phrase has length exactly 1 (i.e. $n_1 = 1$.) The last phrase may or may not be distinct from the previous phrases. Also, for every phrase in $x[1\ldots n]$, every one of its proper prefixes appear in the parsing.

Let L_j be the length of the j^{th} phrase, $j = 1, \ldots, p+1$. The total number of bits when coding an input string $x[1\ldots n]$ into $p+1$ phrases is

$$\begin{aligned}
L &= \sum_{j=1}^{p+1} L_j \\
&= \sum_{j=1}^{p+1} \lceil \log(j\alpha) \rceil \\
&\leq \sum_{j=1}^{p+1} (\log(j\alpha) + 1) \\
&= \sum_{j=1}^{p+1} \log(2j\alpha) \\
&\leq (p+1)[\log(p+1) + \log(2\alpha)] && [j \leq p+1]
\end{aligned}$$

We know that $p \leq c(x[1\ldots n])$, the maximum number of distinct phrases whose concatenation forms $x[1\ldots n]$. Substituting this into the inequality above, we get

$$L \leq C\log(2\alpha C) \qquad \text{where } C = c(x[1\ldots n]) + 1 \tag{1}$$

From this, we get

$$\rho(\mathcal{E}, x[1\ldots n]) \leq \frac{C}{n\log\alpha} \ \log\big(2\alpha C\big), \qquad \text{where } C = c(x[1\ldots n]) + 1$$

This proves (i).

§2. Now, we have to see how well this compressor does, with respect to other compressors. Let s be fixed. Then the converse to the coding theorem gives us that

$$\frac{c(x)+s^2}{n\log\alpha}\log\frac{c(x)+s^2}{4s^2} + \frac{2s^2}{n\log\alpha} \leq \rho(E(s), x).$$

Thus, we can write that

$$\frac{c(x)+1}{n\log\alpha}\log(2\alpha(c(x)+1)) = \frac{c(x)+s^2}{n\log\alpha}\log(2\alpha(c(x)+1)) - \frac{s^2-1}{n\log\alpha}\log(2\alpha(c(x)+1))$$

$$= \frac{c(x)+s^2}{n\log\alpha}\log\left(\frac{2\alpha(c(x)+1)}{c(x)+s^2}\right) - \frac{s^2-1}{n\log\alpha}\log(2\alpha(c(x)+1)) + \frac{c(x)+s^2}{n\log\alpha}\log(c(x)+s^2)$$

We obtain

$$\rho(\mathcal{E},x) \le \rho(E(s),x) + \delta(c(x),n),$$

where

$$\delta(c(x),n) = \frac{c(x)+s^2}{n\log\alpha}\log\left(\frac{2\alpha(c(x)+1)}{c+s^2}\right) - \frac{s^2-1}{n\log\alpha}\log(2\alpha(c(x)+1)).$$

Now, if we use the estimate that for any string $x \in A^n$,

$$c(x) < \frac{n\log\alpha}{(1-\epsilon_n)\log n},$$

where ϵ_n tends to 0 as n tends to ∞, we obtain that

$$\lim_{n\to\infty}\delta(c(x),n) = 0.$$

This proves 2.

§3. The compression attained by our encoder $\mathcal{E}$ for $x \in A^\infty$ is

$$\rho_{\mathcal{E}}(x,n) = \limsup_{k\to\infty}\frac{1}{kn\log\alpha}\sum_{i=1}^{k} L_i,$$

where L_i is the length of the i^{th} block in x of length n.

We know by (1) and (2) that

$$\frac{L_i}{n\log\alpha} = \rho_{\mathcal{E}}(x[(i-1)n+1\ldots in]) \le \rho_{E(s)}(x[(i-1)n+1\ldots in]) + \delta_s(n).$$

Thus,

$$\rho(\mathcal{E},x,n) \le \limsup_{k\to\infty}\frac{1}{K}\sum_{i=1}^{k}\rho(E(s),x[(i-1)n+1\ldots in]) + \delta_s(n) \quad = \rho(E(s),x) + \delta_s(n), \tag{2}$$

where $lim_{n\to\infty}\delta_s(n) = 0$.

Since

$$\rho(x) = \lim_{s\to\infty}\rho(E(s),x),$$

we can write

$$\rho(E(s),x) = \rho(x) + \delta'_s(x), \tag{3}$$

where $\lim_{s\to\infty}\delta'_s(x) = 0$.

From the (**??**) and (**??**), we can write that for any $\epsilon > 0$,

$$\rho(\mathcal{E},x,n) \le \rho(x) + \epsilon + \delta_s(n),$$

which proves (3) with $\delta_\epsilon(x,n) = \epsilon + \delta_s(n)$.

This completes the proof of (3).

3 Converse-to-coding theorem

Theorem 3.1. *For every $x \in A^n$,*

$$\rho(E(s), x) \quad \geq \quad \frac{c(x)+s^2}{n\log\alpha}\log\frac{c(x)+s^2}{4s^2} + \frac{2s^2}{n\log\alpha},$$

where $c(x)$ is the maximum number of distinct *phrases whose concatenation forms x.*

Note that this is an individual lower bound, and hence very powerful. The theorem points to this parsing scheme as one of the ways to capture the essence of finite-state compression.

Proof. Given an encoder E having s-states and an input string $x \in A^n$, let

$$x = x[1\dots n_1]x[n_1+1\dots n_2]\dots x[n_{c-1}+1\dots n]$$

be a parsing of x into c distinct phrases. Let

$$c_j \quad = \quad |\{x[n_{i-1}+1\dots n_i] \mid \text{output\_length}(y[n_{i-1}-1\dots n_i]) = j.\}|\,.$$

In the above definition, recall that $y[i]$ can be λ. Since E is a information-lossless compressor of finite order, we know that

$$c_j \leq s^2 2^j.$$

Suppose

$$c = s^2\left(\Delta_k + \sum_{i=0}^{k} 2^i\right) + r, \qquad 0 \leq r < s^2. \tag{4}$$

(Adjusting to form nice sums.) Then we can assume that $c_j = s^2 2^j$, for $0 \leq j \leq k$, and $c_{k+1} = s^2\Delta_k + r$. Let $c_j = 0$ for $j > k+1$.

Hence

$$c \quad = \quad s^2\sum_{j=0}^{k} 2^j + s^2\Delta_k \quad = \quad s^2(2^{k+1}+t),$$

where

$$t = \Delta_k - 1 + \frac{r}{s^2},$$

and

$$\begin{aligned}
\text{output_length}(y) &\geq s^2\sum_{j=0}^{k} j2^j \;+\; (k+1)(s^2\Delta_k + r)\\
&= s^2[(k-1)2^{k+1}+2] \;+\; (k+1)(s^2(\Delta_k + r))\\
&= s^2((k-1)2^{k+1}+t) \;+\; s^2(k+3+2t)\\
&= (k-1)[c+s^2] + 2s^2(t+2).
\end{aligned} \tag{5}$$

By (**??**) and (**??**), we have that

$$L(y) \qquad \geq \qquad (c+s^2)\left(\frac{c+s^2}{4s^2}\right) + \delta, \tag{6}$$

(Determining δ in terms of s^2.) $\square$

3.1 Existence of empirical entropy rate of a sequence

Lemma 3.2. *For every sequence x,*

$$\hat{H}(x) \qquad = \qquad \lim_{m\to\infty} \limsup_{n\to\infty} \hat{H}_m(x[1\ldots n])$$

exists, where $\hat{H}_m(x)$ is the entropy of the 2^m-dimensional empirical probability distribution defined by x on the set of m-long strings.

Proof.

$$(\ell+m)H_{\ell+m}(x) \quad = \quad \frac{-1}{\log\alpha} \limsup_{n\to\infty} \sum_{w\in A^{\ell+m}} P(x[1\ldots n],w)\log P(x[1\ldots n],w)$$

□