CPSC 420 Lecture 16: Today's announcements:

- Examlet 2 on Feb 17 in class. Closed book & no notes
- Reading: Shor's notes on Lempel-Ziv compression `https://math.mit.edu/~shor/PAM/lempel_ziv_notes.pdf`

Today's Plan

- Compression
  - Huffman Coding
  - Lempel-Ziv Compression

Compression [Lempel & Ziv 1978]

$$AAABABBBBAABBBB$$

- Parse input into distinct phrases reading from left to right.
  Each phrase is the shortest string not already a phrase.
  The 0th phrase is $\emptyset$.
- Output i c for each phrase w, where c is the last character of w and i is the index of phrase u where $w = u \circ c$

Let $c(n)$ be the number of phrases created from input of length n.
Let α be the size of the alphabet of characters in input.
Length of output is $c(n)(\log_2 c(n) + \log_2 \alpha)$ bits.

Compression [Lempel & Ziv 1978]

$$|A|AA|B|AB|BB|BA|ABB|BB$$

- Parse input into distinct phrases reading from left to right.
  Each phrase is the shortest string not already a phrase.
  The 0th phrase is $\emptyset$.
- Output i c for each phrase w, where c is the last character of w and i is the index of phrase u where $w = u \circ c$

1	2	3	4	5	6	7	8
A	AA	B	AB	BB	BA	ABB	BB
$0A$	$1A$	$0B$	$1B$	$3B$	$3A$	$4B$	5
00	10	001	011	0111	0110	1001	111

Let $c(n)$ be the number of phrases created from input of length n.
Let α be the size of the alphabet of characters in input.
Length of output is $c(n)(\log_2 c(n) + \log_2 \alpha)$ bits. size of compressed string

Compression [Lempel & Ziv 1978]

How do we show LZ78 is a good compressor?

Empirical Try it on lots of inputs.

Worst case? Average case? Something else?

Worst case

$c(n) = \#\text{phrases} \qquad \alpha = |\text{alphabet}|$ (assume $\alpha = 2$)

Maximize $c(n)$ (make many small phrases)

$$A|B|AA|AB|BA|BB|\ldots$$

The smallest input with **all** phrases of lengths $1, 2, \ldots, k$ has length

$$n_k = \sum_{j=1}^{k} j2^j = (k-1)2^{k+1} + 2 \qquad \left(\lg n_k \approx k\right)$$

For such an input, $c(n_k) = \sum_{i=1}^{k} 2^i = 2^{k+1} - 2$, so $c(n_k) \le \frac{n_k}{k-1}$.

Compression [Lempel & Ziv 1978]

Worst case continued

In fact, for all n between n_k and n_{k+1},

$$c(n) \overset{❶}{\leq} \frac{n_k}{k-1} + \frac{n-n_k}{k+1} \leq \frac{n}{k-1} \overset{❷}{\leq} \frac{n}{\log_2 c(n) - 3}$$

since ❶ to maximize $c(n)$, the first n_k input bits make $\leq c(n_k)$ phrases and the rest make phrases of length $k+1$, and

❷ $c(n) \leq c(n_{k+1}) = 2^{k+2} - 2$.

As we saw, LZ78 compresses inputs of length n to about

$o(n)$

$$c(n) \log_2 c(n) + c(n) \leq n + \underbrace{\frac{4n}{\log_2 c(n) - 3}}_{} = n + O(\frac{n}{\log_2 n}) \text{ bits.}$$

Is this good? yes

Compression [Lempel & Ziv 1978]

Average case

$$\text{Alphabet} = \{a_1, a_2, \ldots, a_\alpha\}$$

Create input $x = a_{x(1)} a_{x(2)} \ldots a_{x(n)}$ by choosing n characters at random, where a_i is chosen with probability p_i.

phrase

Let $Q(x) = \prod_{i=1}^{n} p_{x(i)}$ be the probability of input x.

If LZ78 breaks x into distinct phrases $x = y_1 y_2 \ldots y_{c(n)}$ then

$$Q(x) = \prod_{j=1}^{c(n)} Q(y_j) = \prod_{\ell} \prod_{|y_i|=\ell} Q(y_i)$$

Let c_ℓ be the number of phrases of length ℓ.
Since the y_i's with length ℓ are distinct $\sum_{|y_i|=\ell} Q(y_i) \leq 1$ and

$$\prod_{|y_i|=\ell} Q(y_i) \leq \left(\frac{1}{c_\ell}\right)^{c_\ell} \quad \begin{array}{l}\text{take log}\\ \text{sum over } \ell\end{array} \quad -\log_2 Q(x) \geq \sum_{\ell} c_\ell \log_2 c_\ell$$

$x(1-x)$

Compression [Lempel & Ziv 1978]

Average case continued $Q(x) = \prod_i p_i^{n_i}$ where $n_i = \#a_i$ in $x \approx np_i$

$$\sum_{\ell} c_\ell \log_2 c_\ell \le -\log_2 Q(x) \approx np_i \log_2(1/p_i) = nH(X)$$

$\geq$

times char a_i shows up

where X is a random character.

Recall, LZ78 compresses to approx $c(n)\log_2 c(n)$ bits. If this is approx $\sum_\ell c_\ell \log_2 c_\ell$ then LZ78 compresses (nearly) optimally [Source Coding Theorem].

In fact,

mystery

$$nH(X) \ge -\log_2 Q(x) \ge c(n)\log_2 c(n) - O(\log_2(\frac{n}{c(n)}))$$

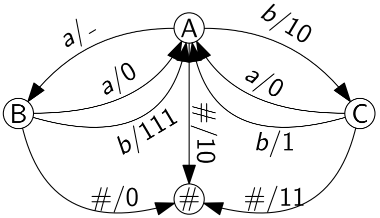
LZ really close to entropy

Compression [Lempel & Ziv 1978]

As good as any finite state compressor

Every text has an algorithm that compresses it well, but the algorithm is as big as the text.

A **finite state compressor** is a finite state machine with output strings on transitions.



input T	Output
$a\#$	0
$aa\#$	010

Require: output of M uniquely determines input T.

1. Show all FSC with s states can't compress better than $r_s(T)$
2. Show $|\text{LZ78}(T)| \leq r_s(T) + o(|T|)$

Compression [Lempel & Ziv 1978]

As good as FSC continued

Let $c(T) =$ max number of distinct phrases T can be split into.
Let $c_j = \#$phrases that cause M to output j bits starting from some state in M.
$c_j \le s^2 2^j$ since [A, j-bit code, B] uniquely specifies phrase x, where A is state when M starts reading x and B is state when it stops.

If two phrases x and y cause the same output going from A to B then M outputs the same encoding for $wx\# \ne wy\#$, where w takes M to state A.

Assume $c_j = s^2 2^j$ for all $j \le k$ i.e. use max number of short codes.

$$c(T) = \sum_{j=0}^{k} c_j \le s^2 \sum_{j=0}^{k} 2^j = s^2(2^{k+1} - 1)$$

Total length of encoding by M:

$$|M(T)| \ge \sum_{j=0}^{k} jc_j = s^2 \sum_{j=0}^{k} j2^j = s^2((k-1)2^{k+1} + 2) = r_s(T)$$

Compression [Lempel & Ziv 1978]

As good as FSC continued

From before

$$|\mathsf{LZ78}(T)| \leq c(T)\log_2 c(T)$$

and

$$r_s(T) = s^2((k-1)2^{k+1}+2) \geq (c(T)+s^2)\log_2(\frac{c(T)}{4s^2})$$

So

$$|\mathsf{LZ78}(T)| \leq r_s(T) + \underbrace{2c(T) - s^2\log_2 c(T) + (c(T)+s^2)\log_2(4s^2)}_{\text{this is } o(|T|)}$$