CHAPTER 3

Universal coding II: pattern matching

3.1. Motivations

Until recently, arithmetic coding and its spin-offs (like CTW) have had little impact on practical text compression programs. This is due to the fact that by the time arithmetic and mixture coding emerged, a computationally simple and asymptotically efficient text compression algorithm was proposed by J. Ziv and A. Lempel (1977, 1978). This algorithm, or rather this family of algorithms was able to compress optimally a very large class of sources in a way that might seem strange at first sight: *pattern-matching,* i.e. by looking for repeated patterns in the string to be coded.

From a computational viewpoint, this is a very appealing technique: it does not require any sophisticated arithmetic, it relies on simple and well-understood data structures called *trie*s, it does not involve any explicit statistical modeling. Moreover LZ-compressed may be searched and processed efficiently.

Hence LZ compression is The main motivation for using the Lempel-Ziv schemes may seem computational but LZ are also fascinating theoretical objects: the proof of first-order optimality and the anlaysis of the redundancy of Lempel-Ziv schemes have been one of the hottest topics in Information Theory. Moreover LZ compression schemes have been an incentive to investigate a large class of text compression methods based on Gammar Transforms.

3.2. The entropy rate of stationary source

Till now, we have considered either very general sources or very specific sources (memoryless sources, Markov sources). This was not a matter of concern, since we were considering encoding of words with fixed length, say n,in this Chapter, we will have to abandon this fixed horizon setting. It will make sense to focus on large, yet restrictde class of sources. The largest class of sources we will consider is the class of stationary sources.

DEFINITION 3.2.1. [STATIONARY SOURCE]

Let P denote the probability distribution of the E-valued random variable X where E is a countable set. The (Shannon) entropy of P is denoted by $H(P)$ and defined by:

$$H_b(P) = \mathbb{E}_P[-\log_b P(X)] \ ,$$

where $\log_b$ denotes the base b logarithm.

It is convenient to take b to be cardinality of the encoding alphabet $\mathcal{Y}$.

For the sake of concision, we will often denote by $H(X)$ the entropy of the distribution of X. Then we have the following corollary of Theorem **??**:

COROLLARY 3.2.2. *If R is an achievable compression ratio for source $(X_n)_{n\in\mathbb{N}}$ then:*

$$R \geq \limsup_n \frac{H_{|\mathcal{Y}|}(X_1^n)}{n}.$$

Note that

$$\limsup \frac{H(X_1^n)}{n}$$

is well-defined and finite: $\frac{H(X_1^n)}{n}$ is positive and smaller than $\log|\mathcal{X}|$. The subadditivity of the entropy actually implies that $\frac{H(X_1^n)}{n}$ has a limit when the source is stationary. This follows from the following Proposition 3.2.3 and Lemma 3.2.4.

PROPOSITION 3.2.3. [SUBADDITIVITY OF ENTROPY]*Let X, Y be two discrete random variables,*

$$H(X,Y) \leq H(X) + H(Y) \ ,$$

equality is only possible when X and Y are independent.

The following simple Lemma proves to be frequently useful.

LEMMA 3.2.4. [FEKETE, CIRCA 1923] *Let f be a function from $\mathbb{N}$ to $\mathbb{R}$ assume:$f(m+n) \leq f(m) + f(n)$ for all $m, n \in \mathbb{N}$, if $\inf_n f(n)/n > -\infty$ then $\lim_n f(n)/n$ exists and:*

$$\lim \frac{f(n)}{n} = \inf \frac{f(n)}{n} < \infty.$$

PROOF. Assume $b = \inf \frac{f(n)}{n} < \infty$, fix $\epsilon > 0$ then there exists some n_0 such that $|\frac{f(n_0)}{n_0} - b| < \epsilon$. Let n be larger than n_0, then $n = kn_0 + r$ with $r < n_0$,

$$\begin{aligned} f(n) &\overset{(a)}{\leq} kf(n_0) + f(r) \\ &\overset{(b)}{\leq} kn_0 b + kn_0\epsilon + f(r) \\ &\overset{(c)}{\leq} nb + n\epsilon - rb + \max_{u<n_0} f(u), \end{aligned}$$

where (a) comes from the subadditivity property of f, (b) from the definition of n_0, and (c) is elementary. This entails that $\limsup_n \frac{f(n)}{n} \leq b + \epsilon$. As ϵ may be chosen arbitrarily small, $\limsup \frac{f(n)}{n} = \liminf \frac{f(n)}{n}$.

The case $\liminf \frac{f(n)}{n} = \infty$ is impossible since we have $f(n) \leq nf(1)$ by subadditivity. □

The lower bound on $f(n)/n$ is actually not necessary. Subaddtivity became part probabilistic consciousness in the sixties with Kingman's subadditive ergodic theorem.

THEOREM 3.2.5. [ENTROPY RATE OF A STATIONARY SOURCE] *Let P define a stationary source. Then the limit $H_\infty(P) \triangleq \lim_n \frac{H(X_1^n)}{n}$ exists and is called the* entropy rate *of the source.*

3.3. The Lempel-Ziv compression schemes (outline)

3.3.1. History. There exists actually two main variants of the Lempel-Ziv schemes. The first one LZ77 ultimately gave birth to `gzip` while the second one LZ78 and its variant LZW rapidly gave birth to `compress`, is used in modems for telephone lines (V42.bis and beyond) and in other compression standards like GIF. Both variants rely on the concept of *parsing.*

3.3.2. Parsings.

DEFINITION 3.3.1. [DISTINCT PARSING] Let $w = w_1^n$ denote a word on some alphabet $\mathcal{X}$. Then $u_1, u_2, \ldots u_m$ with $u_i \in \mathcal{X}^*$ is a *distinct parsing* of w if the concatenation of u_i equals w, i.e. $w = u_1 u_2 \ldots u_m$ and $i \neq j$ implies $u_i \neq u_j$. LZ compression schemes first construct a parsing.

3.3.3. Coding (and decoding) a parsing.

3.4. The LZ78 parsing scheme

3.4.1. Outline. Let us describe the LZ78 parsing process. Assume w is an infinite string. The parsing of the empty string is reduced to the empty string. Assume we have already parsed the first n symbols of w, then we should search the longest prefix of w_{n+1}^{∞} that occurs in the parsing of w_1^n, assume it is u_j, and let $h \triangleq \ell(u_j)$ then the next word in the parsing is w_{n+1}^{n+1+h}.

EXAMPLE 3.4.1. Let the data be constituted by the following 20 binary symbols

$$01010011010000110010.$$

The LZ78 parsing of that string is:

$$\epsilon \quad 0 \quad 1 \quad 01 \quad 00 \quad 11 \quad 010 \quad 000 \quad 110 \quad 010.$$

To encode w it is enough to encode the sequence of blocks that constitute the parsing. Note that the i^{th} block is the concatenation of a previously occurring block and a single symbol. In the example the 7^{th} block 010 is the concatenation of the the 4^{th} block 01 and the single symbol 0. If $u_i = u_j a$ where u_j is a previously occurring block in the parsing, and a a symbol from $\mathcal{X}$, u_i is uniquely defined by $\langle j, a\rangle$, it may thus be encoded using $\lfloor \log_2 j + 1 \rfloor + \log_2 |\mathcal{X}|$ bits. Note that we do not try to optimize the encoding of single letters. The LZ78 encoding replaces a substring with a pointer to where it occurred previously.

$$\epsilon \quad \langle 1,0\rangle \quad \langle 1,1\rangle \quad \langle 2,1\rangle \quad \langle 2,0\rangle \quad \langle 3,1\rangle \quad \langle 4,0\rangle \quad \langle 5,0\rangle \quad \langle 6,0\rangle \quad 010.$$

THEOREM 3.4.2. *The length of the* LZ78 *encoding of a string is thus uniquely determined by the number of substrings constituting the* LZ *parsing. Let* $c(w)$ *denote this number, then:*

$$\begin{aligned} \ell(\mathit{lz}(w)) &\leq \sum_{i=1}^{c(n)} \lceil \log_2(i+1) \rceil + c(n) \lceil \log_2 |\mathcal{X}| \rceil \\ &\leq c(n) \log_2[c(n)+1] + c(n) \lceil \log_2 |\mathcal{X}| \rceil. \end{aligned} \tag{3.4.1}$$

3.4.2. Implementation issues (tries). Before proceeding to the proof of the optimality of the Lempel-Ziv scheme, let us look at the way encoding and decoding could be implemented efficiently. Assume that the first m symbols in w have been parsed, to encode w_{m+1}^n, we need to determine the longest prefix of w_{m+1}^n which occurs in the parsing of w_1^m. Ideally, we should spend a constant amount of time per symbol, in order to achieve LZ compression in time proportional to the length of the string that has to be encoded. This is feasible using digital search trees (also known as *tries*). A digital search tree represents a collection of strings on a given alphabet. A tree reduced to a root represents the empty

string. The tree is assumed to be oriented, edges pointing towards the leaves. Each edge is labelled by an alphabet symbol. All edges outgoing from a node should have distinct labels. Each node is associated with the word defined by the labels on the (unique simple) path starting from the root and reaching that node. Note that any prefix of a string occurring in a LZ78 parsing also occurs in the parsing, hence the digital search tree has one node per block in the parsing. If furthermore each node is labelled by the index of the corresponding block, the digital search tree is a concise representation of the parsing. Let us call it a parse tree.

When starting to parse w_{m+1}^n, keep a pointer to the root of the parse tree, if no edge outgoing from the pointed node is labelled by w_{m+1} then w_{m+1} is a new symbol, and the corresponding block is $\langle \epsilon, w_{m+1} \rangle$, add a new successor to the root in the parse tree and label the corresponding edge by w_{m+1}, iterate the procedure on w_{m+2}^n. If there is an edge outgoing from the pointed node that is labelled by w_{m+1} , move the pointer to the extremity of that edge and iterate the procedure on w_{m+2}^n.

This procedure uses linear time and space. Moreover all operations are elementary, they do not involve any multiplication and division. This is one of the reasons that makes Ziv-Lempel so popular: simplicity.

3.4.3. Decoding.

3.5. The LZ77 parsing scheme

3.5.1. Outline. In the LZ 77 parsing scheme, assuming that text $x_1, \ldots, x_n$ has been parsed up to position m, then the next block of the parsing is constructed in the following way: let $x_{m+1}, \ldots, x_{m+j}$ denote the longest prefix of $x_{m+1}, \ldots, x_n$ that occurs somewhere (not necessarily as a block) in $x_1, \ldots, x_m$. Then the next block in the parsing is $x_{m+1}, \ldots, x_{m+j+1}$ (the text is assumed to be terminated by a sentinel character).

EXAMPLE 3.5.1. Let the data be constituted again by the following 20 binary symbols

$$01010011010000110010.$$

The LZ77 parsing of that string is:

$$\epsilon \quad 0 \quad 1 \quad 010 \quad 011 \quad 01000 \quad 01100 \quad 010.$$

3.5.2. Coding (and decoding). In order to code a block $x_{m+1}, \ldots, x_{m+j+1}$ in an LZ77 parsing, we have to encode three things, the position k of the first occurrence of $x_{m+1}, \ldots, x_{m+j}$ in $x_1, \ldots, x_m$, the length j of this subword and the character x_{m+j+1}, that is, two integers smaller than the total length of the word n, and a character from a fixed alphabet $\mathcal{X}$.

Actually the two integers may be bounded in a less conservative way. Hence coding a block requires at most

$$2\left(\log_2(n) + 1 + 2\log_2\left(\log_2(n) + 1\right) + 2\right) + \log_2|\mathcal{X}|$$

bits. This is a conservative estimate.

3.5.3. Implementation (suffix trees). In order to compute the LZ77 parsing efficiently we need to represent $x_1, \ldots, x_m$ using a data structure the subwords of $x_1, \ldots, x_m$ that enjoys the following properties:

(1) Searching for the longest prefix of a string $y_1, \ldots, y_m$ that occurs in $x_1, \ldots, x_m$ should take time proportional to the length of this longest prefix.
(2) The data structure representing the subwords of $x_1, \ldots, x_m$ should be easy to update. Constructing the data structure associated with $x_1, \ldots, x_{m+1}$ from the data structure associated with $x_1, \ldots, x_m$ should take *constant amortized time*.

As the collection of subwords of a given word is somewhat more complicated than the prefix-closed collection of words handled by the LZ78 parser, we will not be abale to use a trie to fullfill this set of requirements. A collection of related data structures called compact suffix trees or Patricia trees will prove convenient.

DEFINITION 3.5.2. The compact suffix tree associated with word $x_1, ..., x_n$ is a rooted tree where
i) each leaf is associated with a suffix of $x_1, ..., x_n$;
ii) every internal node has at least two children;
iii) each edge is labelled by a subword of $x_1, ..., x_n$;
iv) the concatenation of labels along the branch leading to some leaf gives the suffix associated with that leaf.
v) the subwords labelling two edges outgoing from one node must start by different symbols.

DEFINITION 3.5.3. Every suffix $x_m, ..., x_n$ of $x_1, ..., x_n$ is parsed into two subwords *head* and *tail*. The subword *head* is the longest prefix of $x_m, ..., x_n$ that occurs in $x_1, ..., x_{m-1}$. And *tail* is defined by $x_m, ..., x_n = \mathit{head}\,\mathit{tail}$.

NOTE 3.5.4. Note that when constructing the LZ77 parsing of some word $x_1, ..., x_n$, once we have parsed $x_1, \ldots, x_{m-1}$, we have to find out what are *head* and *tail* in $x_m, \ldots, x_n$.

LEMMA 3.5.5. *In the compact suffix tree associated with $x_1, ..., x_n$ there are at most $2n-1$ nodes.*

This reflects the fact that when building a sequence of partial suffix trees, Mac-Creight algorithm adds at most two nodes when adding a new suffix: one leaf is added and at most one internal node is splitted.

Suffix links enable the efficient construction of suffix trees.

3.6. The regret of Lempel-Ziv scheme with respect to Markov models

Despite their simplicity, Lempel-Ziv coding schemes achieve optimal compression ratio.

THEOREM 3.6.1. [WYNER,ZIV,LEMPEL] *If P defines a stationary source $(X_n)_{n\in\mathbb{N}}$ with entropy rate H_∞, then:*

$$\lim_n \mathbb{E}_{P^n}\left[\frac{\ell[lz(X_1^n)]}{n}\right] = H_\infty.$$

The proof of Theorem 3.6.1 follows from several technical lemma. Note that it is enough to check that

$$\lim_n \mathbb{E}_{P^n}\left[\frac{\ell[lz(X_1^n)]}{n}\right] \leq H_\infty\,.$$

As we already have a nice upper bound on $\frac{\ell[lz(X_1^n)]}{n}$ thanks to inequality (3.4.1), the main objective of the technical Lemmas will be to relate the typical value of $\log_2 P(X_1^n)$ and the number $c(n)$ of blocks in the LZ78 (or the LZ77) parsing of X_1^n.

The first Lemma is elementary and upper bounds the number of blocks in a distinct parsing.

LEMMA 3.6.2. *The number $c(n)$ of blocks in a distinct parsing of a word of length n is upper bounded by:*

$$c(n) \leq \frac{n}{(1-\frac{4+\log\log n}{\log n})\log n}$$

PROOF. Let $c_i(n)$ denote number of blocks of length i in the parsing. We have $\sum_i c_i(n) = c(n)$ and $\sum_i c_i(n) \times i = n$. $c(n)$ is maximized when all possible small blocks are used, i.e. when there are 2^i blocks of size i in the parsing for all small enough i. Let J denote the largest integer such that

$$\sum_{i=1}^{J} i\, 2^i \leq n,$$

i.e. J satisfies:

$$2 + 2^{J+1}(J-1) \leq n < 2 + 2^{J+2}J,$$

then any distinct parsing is constituted by at most

$$\sum_{i=1}^{J} 2^i + [n - 2 - 2^{J+1}(J-1)]/(J+1) \leq 2^{J+1} - 2 + \frac{n-2}{J+1} - 2^{J+1} + \frac{2^{J+2}}{J+1}$$

blocks, i.e.:

$$\begin{aligned} c(n) &\leq 2^{J+1}(\frac{2}{J+1}) + \frac{n-2}{J+1} \\ &\leq \frac{n-2}{J+1}(1 + \frac{2}{J-1}), \end{aligned}$$

as $J \geq \log(n-2) - \log\log(n-2) - 2$ for $n \geq 2$,

$$\begin{aligned} c(n) &\leq \frac{n}{\log(n-2) - \log\log(n-2) - 1}(1 + \frac{4}{\log(n-2)}) \\ &\leq \frac{n}{\log n(1 - \frac{2+\log\log n}{\log n})} \ . \end{aligned}$$

□

The second Lemma provides an example of a series of very useful statements of the following form: among the distribution which satisfy certain moment conditions (i.e. linear constraints) which one do have the largest entropy? Many questions in Information Theory, Statistics, Probability and Statistical Mechanics have such a flavor.

LEMMA 3.6.3. [MAXIMUM ENTROPY]*Among the probability measures on* $\mathbb{N}^+$ *which have expectation* μ*, the law which has maximal entropy is the geometric law with expectation , this entropy is upper bounded by*

$$(\mu+1)\log(\mu+1) - \mu\log\mu$$

PROOF. W.l.o.g. we assume that $\mu > 1$, otherwise the question is trivial. Assume $\mathbb{Q}$ is the geometrical distribution with mean μ and $\mathbb{P}$ is another distribution over the integers with mean μ. By definition of the geometric law:

$$\mathbb{Q}\{k\} = (1 - \frac{1}{\mu})^{k-1}\frac{1}{\mu}.$$

and

$$\begin{aligned} H(\mathbb{Q}) &= -\sum_k (1-\frac{1}{\mu})^{k-1}\frac{k-1}{\mu}\log(1-\frac{1}{\mu}) + \log\mu \\ &= \mu\log\mu - (\mu-1)\log(\mu-1) \\ &\leq (\mu+1)\log(\mu+1) - \mu\log\mu \ . \end{aligned}$$

Now recall that the relative entropy is positive:

$$\begin{aligned} D(\mathbb{P}\,\|\,\mathbb{Q}) &\overset{(a)}{=} -H(\mathbb{P}) - \sum_k \mathbb{P}\{k\}(k-1)\log(1-\frac{1}{\mu}) + \log\mu \\ &\overset{(b)}{=} -H(\mathbb{P}) - \sum_k \mathbb{Q}\{k\}(k-1)\log(1-\frac{1}{\mu}) + \log\mu \\ &\overset{(c)}{=} -H(\mathbb{P}) + H(\mathbb{Q}), \end{aligned}$$

where equality (c) comes from the fact that $\mathbb{Q}$ and $\mathbb{P}$ have the same expectations. □

To finish the proof of Theorem 3.6.1, we will resort to the k-Markovization device that was already used during the proof of the AEP (Theorem **??**). Let $\mathbb{P}^{(k)}$ be defined by:

$$\mathbb{P}^{(k)}(x_{-(k-1)}^n) \triangleq P(x_{-(k-1)}^0)\prod_{i=1}^n P(x_i \mid x_{i-k}^{i-1}).$$

For any block u in a distinct parsing of w, let the state of u be defined as the sequence of k symbols that precede block u in w. For any length l and pattern s of k symbols, let $c_{l,s}$ denote the number of blocks of length l that occur in state s in the parsing of w. Obviously $\sum_{s,l} c_{l,s} = c$ and $\sum_{s,l} c_{l,s} \times l = n$.

LEMMA 3.6.4. [WYNER & ZIV] *For any distinct parsing of the string x_1^n, we have:*

$$-\log\mathbb{P}^{(k)}(x_1^n \mid x_{-k+1}^0) \geq \sum_{s,l} c_{l,s}\log c_{l,s}.$$

PROOF. Let $u_1, u_2 \dots u_c$ denote a distinct parsing of x_1^n. Let $s_1 \dots s_c$ denote the corresponding states ($s_1 = x_{-k+1}^0$), and S the set formed by those states. Let U(s) denote the set of blocks that occur in state s, and $U(s,l)$ denote the set of blocks of length l that occur in state s.

$$\begin{aligned}
-\log \mathbb{P}^{(k)}\{x_1^n\} &= -\sum_{i=1}^{c} \log \mathbb{P}^{(k)}\{u_i \mid s_i\} \\
&\overset{(a)}{=} -\sum_{s\in S} \sum_{u\in U(s)} \log \mathbb{P}^{(k)}\{u \mid s\} \\
&\overset{(b)}{=} -\sum_{s\in S}, \sum_{l} c_{l,s} \sum_{u\in U(s,l))}, \frac{1}{c_{l,s}} \log \mathbb{P}^{(k)}\{u \mid s\} \\
&\overset{(c)}{\geq} -\sum_{s\in S} \sum_{l} c_{l,s} \log \sum_{u\in U(s,l)} \frac{1}{c_{l,s}} \mathbb{P}^{(k)}\{u \mid s\} \\
&\overset{(d)}{\geq} -\sum_{s\in S} \sum_{l} c_{l,s} \log \frac{1}{c_{l,s}} \ .
\end{aligned}$$

where (a), (b) come from the definition of $U(s)$ and $U(s,l)$, (c) comes from Jensen inequality applied to uniform probability on $U(s,l)$ for each couple s, l, and (d) from the fact that $\sum_{u\in U(s,l)} \mathbb{P}^{(k)}\{u \mid s\} \leq 1$ since all u_i are distinct. □

The theorem now follows:

PROOF. Given a distinct parsing $(u_1, \dots u_c)$ of x_1^n such that there are $c_{l,s}$ blocks of length l in state s in the parsing, it is possible to define two integer-valued random variables L and S such that $\mathbb{P}\{L = l \wedge S = s\} = c_{l,s}/c$. S may take at most 2^k values, hence $H(S) \leq k$. And the subadditivity of entropy ensures that $H(L,S) \leq H(L \mid S) + k \leq H(L) + k$. As L takes values in $\mathbb{N}^+$ and has expectation n/c, by Lemma 3.6.3,

$$H(L) \leq \frac{n}{c} \log \frac{n}{c} - (\frac{n}{c} - 1) \log(\frac{n}{c} - 1).$$

$$\begin{aligned}
\sum_{s,l} c_{l,s} \log\, c_{l,s} &= c \log c + c \sum_{s,l} \frac{c_{l,s}}{c} \log \frac{c_{l,s}}{c} \\
&= c \log c - cH(L,S) \\
&\overset{(a)}{\geq} c \log c - ck - cH(L) \\
&\overset{(b)}{\geq} c \log c - ck - n \log \frac{n}{c} + (n-c) \log(\frac{n}{c} - 1) \\
&\overset{(c)}{\geq} c \log c - ck - c \log(\frac{n}{c} - 1) - 2c \ln 2
\end{aligned}$$

where (a) comes from subadditivity of entropy and trivial bounds on $H(S)$, (b) comes Lemma 3.6.3 and (c) from the fact that $x \log(1 - 1/x) \geq -2 \ln 2$ for $x \geq 2$. The Lemma now follows from bounds on c/n (Lemma 3.6.2) □

Note that from the proof, we get the following corollary:

COROLLARY 3.6.5. *For any fixed k, the maximum pointwise regret of the Ziv-Lempel scheme on strings of length n with respect of Markov chains of order k is $O(n \log \log n / \log n)$.*