INFO-H-422 2016-2017

INFORMATION AND CODING THEORY

Exercise Sheet 4

Exercise 1. Lempel-Ziv code.

(a) Consider a source with alphabet {A, B, C, _}. Encode the sequence AA_ABABBABC_ABABC with the Lempel-Ziv code. What is the number of bits necessary to transmit the encoded sequence? What happens at the last ABC? What would happen if the sequence were AA_ABABBABC_ACABC?

(b) Consider a source with alphabet {A, B}. Encode the sequence ABAAAAAAAAAAAAAABB with the Lempel-Ziv code. Give the number of bits necessary to transmit the encoded sequence and compare it with a naive encoding.

(c) Consider a source with alphabet {A, B}. Encode the sequence ABAAAAAAAAAAAAAAAAAAAA with the Lempel-Ziv code. Give the number of bits necessary to transmit the sequence and compare it with a naive encoding.

(d) The sequence below is encoded by the Lempel-Ziv code. Reconstruct the original sequence.
(0 , A), (1 , B), (2 , C), (0 , _), (2 , B), (0 , B), (6 , B), (7 , B), (0 , .).

Exercise 2. We are given a set of n objects. Each object in this set can either be faulty or intact. The random variable X_i takes the value 1 if the i-th object is faulty and 0 otherwise. We assume that the variables $X_1, X_2, \cdots, X_n$ are independent, with $\text{Prob}\{X_i = 1\} = p_i$ and $p_1 > p_2 > \cdots > p_n > 1/2$. The problem is to determine the set of all faulty objects with an optimal method.

(a) How to find the optimum sequence of yes/no-questions that identifies all faulty objects?

(b) – What is the last question that has to be asked in the worst case (i.e., in the case when one has to ask the most number of questions)?
– Which two sets can be distinguished with this question?

Exercise 3. Alice chooses an object from a set and Bob tries to identify the chosen object with a series of yes/no questions. We assume that Bob uses an optimal code (which has the minimum expected length) with respect to the probability distribution from which Alice choses the object. We observe that on average, Bob needs to ask 35 questions to identify the object chosen by Alice. Find a lower bound on the number of objects in the set.

Exercise 4. There are n coins, where one of them may be counterfeit. A counterfeit coin has a different weight from an authentic one. In order to find the counterfeit coin, all coins are weighed.

(a) What is the probability distribution of the counterfeit coin that maximizes the Shannon entropy? Treat the existence or non-existence of a counterfeit coin as an extra element in the probability distribution.

(b) Establish the equation that bounds from below and above the expected number of weighings, if the optimal method is used.

(c) Deduce the maximal number of coins n, given an average number of weighings k.

(d) Show that this maximum can be attained.

Exercise 5. A random variable X can take m possible values and has entropy $H(X)$. Let C be an instantaneous code for the source X, with the expected length $L(C) = H(X)/\log_2 5 = H_5(X)$.

(a) Show that each symbol of X occurs with probability of the form 5^{-n}, where n is a positive integer.

(b) Show that m has to satisfy $m = 4k + 1$ where k is a positive integer.

The exercises and solutions are available at `http://quic.ulb.ac.be/teaching`

1 Background

Abraham Lempel and Jacob Ziv invented two data compression codes, often referred to as LZ77 and LZ78, named after the year each code appeared [1, 2]. Here, the "Lempel-Ziv code" will be referring exclusively to LZ78.

The Lempel-Ziv is a source coding technique with the following advantages:

- The compression is universal, which means the probability distribution of the source symbols does not need to be known in advance.
- The code is asymptotically optimal, i.e. the number of bits tends to the entropy.
- The code is easy to implement as an algorithm.

2 Principle

The source sequence (or string) is divided into substrings of minimum length which did not occur before. For example, 1011010100010 is divided into $1, 0, 11, 01, 010, 00, 10$. Each time a substring needs to be extracted, a sufficient amount of memory is needed to store previously extracted substrings to ensure there is no repetition. For 010 in our example, 0 exists already, so does 01. As a result, 010 is the shortest substring that has not occurred before. Note that each substring has as prefix a previous substring. If this were not the case, the extracted substring would not be the shortest.

Once the sequence is split, the Lempel-Ziv code transforms each substring into a pair comprising:

1. The position of the longest substring which is the prefix of the current substring.
2. The suffix that makes the current substring unique.

For instance, the substrings $1, 0, 11, 01, 010, 00, 10$ are encoded as $(0, 1)$, $(0, 0)$, $(1, 1)$, $(2, 1)$, $(4, 0)$, $(2, 0)$, $(1, 0)$. By convention the substrings are numbered from left to right, starting from 1. The substring at position 0 is the empty string.

In other words, the Lempel-Ziv code transforms a sequence of elements of an alphabet $\mathcal{A}$ into a new sequence of elements belonging to $\mathbf{Z} \times \mathcal{A}$. In the example above the alphabet is binary.

As noted above, the position also needs to be encoded (it can be in an arbitrary alphabet $\mathcal{A}$). The longer the source sequence, the more bits are needed to encode the position. Fortunately, it can be verified that asymptotically the number of bits per symbol tends to the entropy of the source.

Also note that the fundamental disadvantage of the Lempel-Ziv code is the need for a possibly big memory in order to store all the extracted substrings. And this memory is not bounded asymptotically.

3 Implementation

In order to encode using the Lempel-Ziv code, a dictionary that contains all past substrings is needed. Then the algorithm below is used:

1. The dictionary is initialized by putting the empty string at position 0.
2. Extract a substring of $n + 1$ bits[1]. This substring has to be the shortest substring which is not present in the dictionary, where the first n bits correspond to an existing entry of the dictionary.
3. A code is generated which represents the position of the entry in the dictionary and the $(n + 1)$-th bit of the substring, as obtained in the previous step.

[1] Bits are used for simplicity. In general, it is easy to generalize the method to any other alphabet.

4. The extracted substring is added to the dictionary. The position of the entry in the dictionary corresponds to the first empty entry[2].

5. Repeat steps 2 to 4 until the entire source sequence is encoded.

Comments:

- In order to properly treat the end of the sequence, a special character is needed at the end. We shall use dot "." in this exercise.
- The decoding can be directly deduced from the encoding. It suffices if the decoder updates its dictionary in the same way as the encoder.

In our example with the sequence `1011010100010`, the algorithm works as follows:

Substring	Output	Dictionary
		`dico[0] = ""`
`1`	$(0,1)$	`dico[1] = "1"`
`0`	$(0,0)$	`dico[2] = "0"`
`11`	$(1,1)$	`dico[3] = "11"`
`01`	$(2,1)$	`dico[4] = "01"`
`010`	$(4,0)$	`dico[5] = "010"`
`00`	$(2,0)$	`dico[6] = "00"`
`10`	$(1,0)$	`dico[7] = "10"`

4 In theory...

For a detailed theoretical explanation, it is helpful to read Section 13.4 of [3]. We state here the final result, which proves the asymptotic optimality of the Lempel-Ziv code. For a stationary and ergodic source $\{X_i\}_{-\infty}^{+\infty}$ it can be proven that

$$\limsup_{n\to\infty} \frac{1}{n} l(X_1, X_2, \ldots X_n) = H(\mathcal{X}) \quad \text{with probability 1,} \tag{1}$$

where $l(X_1, X_2, \ldots X_n)$ is the length of a codeword of the Lempel-Ziv code associated with $X_1 \ldots X_n$, where $H(\mathcal{X})$ is the entropy of the source.

5 In practice...

In practice, the variant LZW due to Lempel, Ziv and Terry Welch [4] is the most widely used. Examples include the command `compress` in Unix and the image format GIF. [5] gives much more detail.

References

[1] J. Ziv A. Lempel. A universal algorithm for sequential data compression. *IEEE Transaction on Information Theory*, 23(3), 1977.

[2] J. Ziv A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transaction on Information Theory*, 24(5), 1978.

[3] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory. 2nd Ed.* Wiley, 2006.

[4] Terry Welch. A technique for high-performance data compression. *IEEE Computer*, 17(6), 1984.

[5] Mark Nelson *The Data Compression Book.* M&T Publishing, 1992.

[2] Note that here the number of the entry in the dictionary is equal to the number of appearance of the substring, in other words its position