TEL AVIV UNIVERSITY אוניברסיטת תל-אביב
FACULTY OF ENGINEERING הפאקולטה להנדסה

Department of Electrical Engineering – Systems

Universal Finite Memory Coding of Binary Sequences

Thesis submitted towards the degree of
Master of Science in Electrical and Electronic Engineering
in Tel-Aviv University

by

Doron Rajwan

December 2000

TEL AVIV UNIVERSITY אוניברסיטת תל-אביב
FACULTY OF ENGINEERING הפאקולטה להנדסה

Department of Electrical Engineering – Systems

Universal Finite Memory Coding of Binary Sequences

Thesis submitted towards the degree of
Master of Science in Electrical and Electronic Engineering
in Tel-Aviv University

by

Doron Rajwan
e-mail: doron_rajwan@yahoo.com

This research work was carried out at Tel-Aviv University
in the Department of Electrical Engineering - Systems,
Faculty of Engineering,
under the supervision of Prof. Meir Feder

December 2000

Acknowledgments

This whole work was made possible due to the devoted guidance and support of Prof. Meir Feder, my supervisor. His enthusiasm and ideas inspired me during this research, and I thank him for that.

Also, I would like to thank my family for their love and support, specially, my wife, Ofira.

Abstract

This work considers the problem of universal coding of binary sequences, where the universal encoder has limited memory.

Universal coding refers to a situation where a single, universal, encoder can achieve the optimal performance for a large class of models or data sequences, without knowing the model in advance, and without tuning the encoder to the data. In the previous work on universal coding, specific universal machines, whose performance attained the theoretical limits, were suggested. However, these machines require unlimited amount of memory. This work investigates the case where the universal machines are with limited resources, i.e., finite-state machines.

To simplify the problem, this work considers universal finite-state machines that assign probabilities to the next bit. It ignores the additional number of states needed to translate the assigned probabilities into code bits.

In most cases examined, this work provides lower bounds on the performance, which describe the optimal rate (as a function of number of states) that the entropy, or the empirical entropy, can be attained. In addition, in most cases, this work presents specific machines whose performance is compared to these bounds.

While the general problem of universal coding with limited resources is still open, this work provides a set of basic results that will be useful in analyzing the general problem. These results thus provide an important step in understanding the problem of limited resources universal coding.

Contents

List of Figures

Chapter 1

Introduction

This work considers the problem of universal coding of binary sequences, where the universal encoder is a finite-state machine. More accurately, the problem considered in this work is universal probability assignment for the next outcome of a binary sequence, under the self-information loss. The self-information is the ideal codelength, associated with the assigned probability model, for encoding the next outcome. Thus, ignoring the number of states required to convert the assigned probability to code bits, say, by arithmetic encoder, the work essentially considers the universal finite memory lossless coding problem.

In the recent years the universal coding problem has been extensively investigated. As it is well known, optimal lossless coding is achieved by designing an encoder that fits the probability model of the data, to achieve the minimal codelength, i.e., the entropy. Universal coding refers to a situation where a single, universal, encoder can achieve the optimal performance for a large class of models, without knowing the model in advance. In the previous work on universal coding, specific universal machines, whose performance attained the theoretical limits, were suggested. However, these machines re-

quire unlimited amount of memory. This work investigates the case where the universal machines are with limited resources, i.e., finite-state machines.

The universal coding problem can traditionally be presented in two different settings. In the probabilistic setting the data sequence is generated by an unknown probabilistic source, e.g., Bernoulli i.i.d. source or q-th order Markov source with unknown probabilities, and the goal is to attain the source entropy. In the deterministic setting, the data sequence is an arbitrary deterministic individual sequence, and the goal is to attain, e.g., the sequence empirical entropy, which is the minimal codelength of an encoder tuned to this sequence. This work considers both settings, and examine how to attain the source entropy, or the sequence empirical entropy, with a finite-state encoder.

While the work restricts the encoder to be finite-state, it examines various cases where the universal encoder is deterministic, randomized, time-invariant or time-variant. In most cases it provides lower bounds on the performance, which describe the optimal rate (as a function of number of states) that the entropy, or the empirical entropy, can be attained. In addition, in most cases, the work presents specific machines whose performance is compared to these bounds.

Many of the results presented in this work have been summarized in a paper presented in the 2000 Data Compression Conference (DCC) [19].

1.1 Review of Universal Source Coding

Before describing the specific results of the thesis, we present in the following section a review of universal source coding, including an elaboration on the stochastic and deterministic settings, and the relation between data

compression and prediction. A reader familiar with these topics may skip to section 1.2 on page 14.

Source coding

Lossless source coding is a process in which an encoder assigns a sequence of bits to an information source. Later, a decoder, related to the encoder, can decode the sequence of bits into its original form. For example, information source can be a document written in English. This document can be converted into a sequence of bits using ASCII encoding. In this encoding, each letter gets an 8 bit representation. For example, the letter "A" is converted into the bits "01000001". Also, control information, like space between words, end of line, end of paragraph, etc., is converted into bits. Later, the English characters are being decoded by a decoder specialize in converting the bit sequence to English text. These English characters can be displayed on-screen, printed, or even read through the speaker using voice-coder.

Although translation of the characters into bits, without loss of information, is performed by this method, it is not the most effective way to represent the English language. First, the 8-bit sequences does not appear at the same number of times, on average. In almost all English documents the 8 bits representing the letter "e" will appear more than the 8 bits representing the letter "q", which will appear much more than the 8 bits representing a vertical tabulation (a control character which is not in use today). Another inefficiency is that the English letters are not independent. There are dependencies within a single word, e.g., the pair "qu" is more likely to appear than the pair "uq" in English words. Also, there are dependencies between words, because of special linguistic rules applied to the language.

Data compression

Lossless data compression is a method to take a sequence of symbols from a known source, like ASCII encoded English above, and convert it into a more compact form, meaning, less bits on average. From a simple counting argument, this cannot be done for **all** sequences. While some sequences get shorter, others are getting longer. The complicated part is to encode the English bit-stream in such a way that almost all English documents will be shortened. The decoder, naturally, expands the bit sequence from the compact form into the original form, with perfect accuracy.

A simple way to compress this type of data is to use a predefined dictionary. For example, encoder and decoder will use an English dictionary with $32,000$ words (which also contains all the single-letter characters). The encoder will encode each of the words that exist in this dictionary into 15 bits, which is suitable for selecting a specific dictionary entry. This is less than the number of bits allocated for 2 letters in the raw data. Each unknown word will be encoded, letter by letter, into 15 bits per letter, which is more than the original form.

This process has several drawbacks. First, different sectors use different vocabulary. For example, the English language used by an Engineer is different from the English language used by a Doctor (not to mention the hand writing ...). Thus, dictionary specific to each subject will probably contain more than $32,000$ words, causing each word to use more bits in the encoded sequence. It seems that one dictionary cannot fit both.

Second, this process is not capable of learning. For example, where a document is repeatedly using a name, this process will encode it to 15 bits per letter each time, which is way too much. A more efficient way is to add this word to the dictionary.

Third, this process is not robust; it can fail miserably. For example, a document with the letter "x" repeated $1,000,000$ times have a raw size of $8mbit$, and a compressed size of $15mbit$. This means that a document described in a single sentence will have a huge raw size, and will become almost 2 times **bigger** after "compression".

These three examples set an enormous need for a more efficient, learning, coding method.

Universal coding

Universal coding is a coding scheme in which the encoder can accept more than one type of input, leading to a more robust process. A universal encoder can compress efficiently different types of English vocabulary, used by different sectors, and maybe, even other languages.

A simple-to-understand universal encoder can be a combination of several (N) non-universal encoders, in a method known as two-step code. The encoder will first collect all the raw data. Then, it will encode the data using all N non-universal encoders. It will select the encoder with the most compact representation of the data. Then, it will encode it using a two-step code. First, it will transmit the selected encoder, using $\lceil \log_2(N) \rceil$ bits. Then, it will transmit the encoded data, given the selected encoder.

The universal decoder in this case is a trivial extension of the non-universal decoders. First, it detects the decoder to use, from the first $\lceil \log_2(N) \rceil$ coded bits. Then, it passes the rest of the data to the selected (non-universal) decoder.

This basic example demonstrate some well-known properties of **every** universal code. First, each universal code has this tradeoff between features and compactness. In this example, one can enlarge N, thus enable detection

of a more suitable non-universal code, but the increase in flexibility results in a "payment" in a form of larger redundancy for the first part. Second, each universal code can represent only few data types effectively, while it cannot efficiently represent other types. Third, each universal code has some inherent redundancy over a non-universal code that is tuned to a specific type of data.

It turns out that it is possible to define an effective universal code, with the following features:

- It is effective for many data types, or even asymptotically infinite number of data types.
- It has a small, almost zero, normalized redundancy over a non-universal code tuned for each of these data types.
- It is general; not a collection of different codes.
- Data can be encoded on-the-fly, with only one pass, maintaining a relatively small state.
- When it fails, data is not expanded much more than its original size.
- Some codes even have a structure that enables in-depth theoretical inspection.

Entropy, and the divergence distance measure

When the probability of a, say, binary event is known in advance (to both the encoder and decoder), one can design the perfect code for it. Denote p as the probability of a bit x to be one. Also, denote the "codelength" associated for x as l_1 bits if x turns out to be one, and l_0 bits if x turns out to be zero.

Both codelengths must be positive numbers, although they may be a fraction. Following Kraft inequality [13], l_1 and l_0 cannot be arbitrary small:

$$\sum_{i=0}^{1} 2^{l_i} \leq 1. \tag{1.1}$$

In order to have minimal redundancy over the optimal code, a code should at least satisfy this inequality at equality. It means, that one can define some arbitrary parameter, q, which is a number in the range $[0, \ldots, 1]$, and define $l_1 = -\log_2(q)$ bits, and $l_0 = -\log_2(1-q)$ bits, thus Equation (1.1) is satisfied at equality. In this case, the average codelength is:

$$-p\log_2(q) - (1-p)\log_2(1-q). \tag{1.2}$$

Also, define the binary entropy function:

$$h(p) \triangleq -p\log_2(p) - (1-p)\log_2(1-p) \tag{1.3}$$

which is positive because $0 \leq p \leq 1$, and the divergence distance measure between p and q:

$$\mathrm{D}\left(p \parallel q\right) \triangleq p\log_2(\frac{p}{q}) + (1-p)\log_2(\frac{1-p}{1-q}). \tag{1.4}$$

Combining these definitions, the average codelength is the sum of the binary entropy function of the real probability, and the divergence between this probability and the arbitrary parameter q. By proving that $\mathrm{D}\left(p \parallel q\right) \geq 0$, and that it is zero iff $p = q$, we show that the minimal codelength is $h(p)$ bits, and it is achieved by using the codelength which is the self-information associated with p, thus, $l_1 = -\log_2(p)$ bits, and $l_0 = -\log_2(1-p)$ bits. So, the parameter q is actually the probability assigned by the universal coding process, and the extra bit-rate above the entropy is $\mathrm{D}\left(p \parallel q\right)$.

This codelength, although not an integer, is the number of bits that the encoder sends to the decoder for encoding each source bit. For example, if

$p = 0.9$, it is possible to encode a result of $x = 1$ by 0.152 bits, and a result of $x = 0$ by 3.322 bits, which gives 0.469 bits in average. If the universal coding process assumes that $q = 0.89$, the extra bit-rate above the entropy is 0.000757 bits per source bit, and if $q = 0.91$, the extra bit-rate is 0.000859 bits per source bit. Thus, there is some penalty for misdetection of p.

The proof that $\mathrm{D}(p \parallel q) \geq 0$ follows from Jensen inequality:

$$E[\log_2(X)] \leq \log_2[E(X)] \tag{1.5}$$

for any positive random variable X, and equal iff X is a constant. Using this inequality, and defining X as a random variable with value of q/p in probability p and $(1-q)/(1-p)$ in probability $1-p$:

$$\begin{aligned} -\mathrm{D}(p \parallel q) &= p \log_2\left(\frac{q}{p}\right) + (1-p)\log_2\left(\frac{1-q}{1-p}\right) \\ &= E[\log_2(X)] \\ &\leq \log_2[E(X)] \\ &= \log_2\left(p\frac{q}{p} + (1-p)\frac{1-q}{1-p}\right) \\ &= \log_2(1) = 0. \end{aligned} \tag{1.6}$$

Probability assignment

Any lossless coding method is essentially a method for sequential probability assignment. In order to implement an encoder, one can take the assigned probability p_t for the next bit x_t, to be one, given the previous observations $x_1, \ldots, x_{t-1}$, and convert it into code, sequentially. If this assigned probability is the true probability, the resulting code is optimal. On the other hand, any encoder is actually assigning probabilities. If the code for a given sequence of symbols is n bits, it means that the encoder assigned a probability of 2^{-n} for that sequence of bits to appear at this position.

A non-universal encoder has a fixed, predefined probabilities. For example, the probability that ASCII encoder assigns to the letter "A" is 1/256, without regarding the context. A universal encoder, on the other hand, has to learn the probability assignment when it encodes the bits, in order to give a more accurate estimate for the next bit. When the probabilities are known, it is easy to encode the data into bits. This translation can be done by the technique of [12] and other related techniques.

Stochastic setting

What is the "true" probability anyway? Is it possible to compress a sequence of bits to any factor, or, is there a lower limit?

In the stochastic setting, data is modeled as the output of a stochastic process, with a deterministic, predefined set of probabilities. In this case, the "true" probability is the probabilities as defined by the data model. There is a lower limit on the average compression rate of the data – Shannon entropy [1].

If the encoder (and the decoder) knows the data model, it is possible to use a simple, non-universal code. When fed with the right data, this encoder will encode the data perfectly, to its entropy. But, if the data model is incorrect, or inaccurate, this encoder will fail.

Universal encoder, on the other hand, will try to learn the data model, and then assign probabilities sequentially. Meaning, each universal encoder tries to adapt the data to a set of models.

The simplest example is the universal encoder for Bernoulli data models. In these models, data bits are independently identical distributed, with fixed probability $p \triangleq Pr\{x_t = 1\}$. Every universal coding process should estimate this unknown parameter p. For example, it can do it by using the Krichevsky-

Trofimov estimate [9]:

$$\hat{p}_{t+1} = \frac{N_t(1) + \frac{1}{2}}{t+1} \tag{1.7}$$

where $N_t(1)$ is the number of ones in $x_1, \ldots, x_t$. Unlike the two-step code described above, this encoder works on-the-fly, utilizing a single pass over the data. The estimate of p gets more accurate over time, as needed in order to compress efficiently. For example, if the data length is $1,000$ bits, there is no need for the encoder to describe p with more than 3 decimal digits of accuracy. This is exactly what this process does – the encoder implicitly sends p to the decoder with an accuracy of $1/n$.

If the data is Bernoulli, the redundancy of this process, per bit, will converge to zero. But if, for example, this encoder will be fed with the non-Bernoulli data $0, 1, 0, 1, 0, 1, \ldots$, it will not be able to compress it at all! At least it will never expand the data too much.

A stronger universal encoder uses the Markovian model of some fix order q. In this model, probability of each bit depends on previous q bits. Thus, in order to fully describe the data model, learning 2^q independent parameters is needed. A universal encoder (for order q) can assign priorities using Equation (1.7) for each of the 2^q parameters independently. In the above example, if the code is designed for $q \geq 1$, data will be compressed asymptotically to zero code bits per source bit.

This scheme implies setting the Markovian order, q, in the encoder and the decoder before seeing the actual data. This is a complicated task. One advantage of using small q is the fast convergence rate. A disadvantage is the convergence to a sub-optimal point, if the actual data has higher order Markovian elements.

Lempel and Ziv suggested a well-known method [7, 8] that effectively uses higher order of q when collecting more data. This universal encoder is widely

implemented today, being used in programs like **gzip** and others.

Still, it is hard to create a universal encoder that can encode any stochastic sequence. Consider the following: probability of the next bit (x_{t+1}) to be one is 0.01, 0.5 or 0.99, depending on the binary number represented by $(x_1, \ldots, x_t)$ modulo 3. Because this sequence is not Markovian, of any finite order q, it will take a very "heavy" Markovian encoder to encode this sequence efficiently, although it will converge eventually, for high enough q, as proven in [15].

Deterministic setting

A different setting is the deterministic data model. This setting is somewhat more subtle to define and understand. In this setting, data is an arbitrary individual sequence, with a finite or infinite size. For example, Windows NT service pack 7 (SP7), this LaTeX file, or the infinite binary representation of the number π.

Since there is no statistic model for these bit sequences, there is no defined Shannon entropy, and there is no compression lower-limit. When calculating the size of a compressed file, should the size of the decompression program itself be added? Normally, the answer should be no. But then, one can create a decompression program that expands the bits $0, 0, 0$ to SP7 (by integrating SP7 into the decompression program). Does it mean that SP7 is compressible to 3 bits? Counting compression program size, one can create a CPU that when executing a single assembly command it will expand the whole SP7 into its memory. Should the number of transistors in the CPU be counted as well?

The answer was given by Kolmogorov [2]. He defined a quantity, analogous to the entropy, for a deterministic sequences, as the minimal size of a

program that runs on a universal Turning machine, needed to generate this sequence. Thus, the entropy of SP7 is the size of the smallest executable that will expand itself to SP7, when running on a standard Turing machine.

Of curse, in practice, no one can compute the Kolmogorov entropy of SP7. It turned out, that it can not be done even in theory, by running all possible programs and see if their output is SP7, due the **halting problem**. It is impossible to compute Kolmogorov entropy of π as well, but it is easy to prove that it is finite, meaning, that the entropy per bit, of the number π, is zero.

If the process is stochastic, there is a strict relation between Kolmogorov entropy and Shannon entropy – they are (almost) the same, with probability one, for long enough data sequences.

A more feasible method to measure the complexity of deterministic sequences is to check its empirical Markovian entropy of order q. For any of the 2^q histories, one needs to compute the empirical entropy of the next bit. The weighted average of these computation is the empirical entropy of the sequence, of order q. It is easy to prove that the empirical Markovian entropy is a monotonic non-increasing function of q. Also, for any finite sequence, with only N bits, the empirical Markovian entropy will converge to zero when $q \geq N$. This is exactly what happened in the example above of compressing SP7 into 3 bits. On the other hand, when the sequence is infinite, the empirical Markovian entropy will usually not converge to zero. If the process is stochastic and ergodic, there is a strict relation between the empirical Markovian entropy of high enough order ($q \to \infty$) and the Shannon entropy – they are (almost) the same, with probability one, for long enough data sequences.

If π is a normal number[1], its empirical Markovian entropy is maximal, i.e., 1 coded bit per source bit, meaning that it cannot be compressed at all using a finite-state machine. On the other hand, as discussed before, its Kolmogorov entropy per bit is 0, meaning that it is compressible to zero coded bits per source bit, using a Turing machine. This demonstrates, to the extreme, the limitations of the empirical Markovian entropy, of any finite order. Personally, I doubt that any practical encoder can ever break the empirical Markovian entropy limit.

Compression vs. prediction

In the compression problem, or the priorities assignment problem, the encoder has to sequentially assign probabilities for the next bit to be one, with a self-information criterion. In the prediction problem, on the other hand, the predictor has to predict whether the next bit is going to be one or zero, with, say, a criterion of minimal number of errors.

The prediction problem is somehow simpler. For example, in the compression problem the encoder need to distinguish between the cases where the probability for the next bit to be one is 0.123 or 0.124, while in the prediction problem the predictor should predict zero in both cases.

There is no one-to-one relation between the compressibility and predictability of a sequence using a finite-state machine (with number of states $q \to \infty$). Only in the extreme case such a relation exists: a sequence is fully predictable iff it is compressible to zero; a sequence is fully unpredictable iff it is also uncompressible.

As found in [15], the compressibility of a sequence, $p(x)$, is bounded from

[1] A number whose digits are random-like, in any sequence length.

above and below by its predictability, $\pi(x)$, by the equation:

$$2\pi(x) \leq p(x) \leq h\left(\pi(x)\right) \tag{1.8}$$

where $h(\cdot)$ is the binary entropy function, as defined in Equation (1.3). Also, it is noted there that both lower bound and upper bound are achievable. Consider, for example, a sequence that contains 4 zero bits, and then a single random bit, with probability $p = 0.5$ to be one. Consider another sequence that is a Bernoulli with $p = 0.1$. In both cases, predictability is $\pi(x) = 0.1$, meaning that, on average, there are 10% for prediction errors per each bit. The compressibility, however, is different for the sequences. In the first case, compressibility is $p(x) = 0.2$, which is the lower limit. In the second case, compressibility is $p(x) = h(0.1) = 0.469$, which is the upper limit.

1.2 Universal Coding with Limited Resources

In real life the complexity of the coding process is always limited. However, the universal encoders presented above, even in the most simple case where data comes from a Bernoulli process, require unlimited resources. Specifically, the suggested universal coding scheme assigns probabilities according to Equation (1.7). This equation implies that the encoder and the decoder should know t and $N_t(1)$, for every t, meaning that they need to have an infinitely growing memory in order to store these two infinitely-large integer numbers with a perfect accuracy.

In order to overcome the infinite memory requirement for the accurate implementation of Equation (1.7), two straight-forward approximate solutions can be suggested. One solution is to use simple counting when possible, but then freeze both counters when t reaches its upper limit, i.e., when there are no more states to hold bigger values of t and $N_t(1)$. In this solution,

all successive data bits do not modify the estimation $\hat{p}$ of p. This means, however, that if the data model is slightly wrong, and p is slowly changing in time, the encoder cannot track the variations and may fail.

Another solution is to reset both counters to zero when t reaches its upper limit. In this case, the encoder works in "blocks", returning to $\hat{p} = \frac{1}{2}$ from block to block, independent of the data. This encoder will perform better if the model is incorrect, thus, it is more robust, at the expense of its accuracy. Later on in the thesis it will shown that both solutions are far from optimal, and another, better, solution will be presented for this Bernoulli case.

In order to analyze the limited resources coding problem we should first define how to measure the complexity of a coding process. Should the complexity measure be the size of the binary code implementing it on a PC? Should it be the die size, or the number of gates, in hardware implementation? There can be many measurements. In this work, as in other previous works on limited resource universal machines, the complexity limitation is chosen to be limited memory. The machine is constrained to be a finite-state machine with K states.

Previous results – limited memory universal machines

Probably the first set of results on data inference using finite-state machines was obtained by Cover [3] and Hellman and Cover [4, 5, 6]. This work considers hypothesis testing with finite memory. It determined the minimal extra error probability, as a function of the number of states, associated with finite-state limitation.

Later on, Leighton and Rivest considered a problem of assigning a probability to a sequence, using a finite-state machines, where the goal was to approach the true probability that governs the data, with a minimal square

error criterion [11]. These results are extensively utilized in this thesis.

As noted above, universal prediction is closely related to universal coding. The finite memory universal prediction problem has been investigated in [17, 18]. The main results there is the introduction and analysis of two finite memory universal predictors: the sliding window predictor and the saturated counter predictor. Both predictors attain the predictability under the Bernoulli stochastic setting and the single-state predictability for the deterministic setting, but the saturated counter achieves that at a higher rate. The encoders suggested in this thesis have some similarities to these predictors.

1.3 Thesis Outline

We are now ready to present the thesis outline in specific terms. The thesis considers universal coding, or universal probability assignment, of binary sequences, where the universal encoder is constrained to have a finite memory, i.e., it is constrained to be a finite-state machine with K states. It actually considers the problem of universal probability assignment using a finite memory machine. As noted above, the complexity, or the amount of states, needed to translate the assigned probability into code bits, is ignored. This translation to code bits has a known cost per number of states and can be done, e.g., by the technique of [12].

The work considers three different settings for the binary sequences: It can be stochastic or individual sequence; if stochastic it can come from a Bernoulli source, or a q-th order Markov source; if deterministic, the goal is to compete with a single-state reference machine. The universal finite-state machines analyzed in the thesis can be either deterministic or randomized,

and can be either time-variant or time-invariant.

The purpose is to find, for each case, lower and upper bounds on the redundancy, and to specifically describe a machine that attains the upper bound. So far, the behavior in some of the cases is not completely known. Unlike the classical work in universal coding, that analyzes the redundancy as a function of the observation length, this work focuses on the effect of the number of states, K, on the redundancy at a steady-state, i.e., after long enough (infinite) time.

It turns out that, in all the cases considered in this work, the redundancy goes to zero as the number of states goes to infinity. This is not a trivial result; in the prediction problem, only a random machine can achieve it, as shown in [15].

This thesis is organized as follows. In chapter 2, the architecture of the coding system is described, as well as the specification of the various cases considered in the thesis. Then, chapter 3, discusses the optimal way to quantize the probability values that are assigned to states. Chapter 4 analyzes universal K-states encoders for the Bernoulli case, and in chapter 5, this analysis is carried on for the q-th order stochastic Markovian case. Chapter 6, considers the deterministic setting, where the goal (reference) is to compete with a single-state machine tuned to the data. The thesis is concluded at chapter 7.

Chapter 2

Data Settings and Machine Types

The universal coding problem, with limited resources, can be defined at different settings, reflecting assumptions on the data. For example in the stochastic setting the data comes from a unknown stochastic process and the goal is to reach the entropy of that process. In the deterministic setting, the data is arbitrary, but the goal is to reach the performance of a constrained batch encoder. In addition, the problem is defined by the specification of the limited resources universal finite-state machine, which can be deterministic, stochastic, time-invariant or time-variant.

This chapter begins with a schematic description of the architecture of the finite-state coding system analyzed in the thesis. The presented scheme emphasizes that the finite-state machine is used for probability estimation. Then, the chapter summarizes the various cases (data settings and machine types) that is analyzed later in the work. The chapter ends with the definition of the minimum redundancy goal.

2.1 System Architecture

The coding system analyzed in this work consists of an encoder, which compresses the data bits into coded bits, and a decoder, which expands the compressed bits to their original form, with a perfect accuracy. The encoder assigned probabilities using a K-state universal probability estimate, and then translates the actual bit x_t into coded bits, using arithmetic encoder [12]. The decoder decodes the coded bits, using arithmetic decoder, and assigns probabilities for the next bit, sequentially. The block noted by **D** is a single bit delay block. The system architecture is illustrated in Figure 2.1.

The K-state universal probability estimate block is common to the encoder and the decoder. It accepts the data bits, with a single bit delay, i.e., $x_1, \ldots, x_{t-1}$, and assigns probability for x_t to be one. In both places, it is initialized with the same values, and fed with the same bits. Thus, it will provide the same probability estimation, as needed, say, for the arithmetic coding process. Note that when the encoder is a randomized machine, we assume that the encoder and the decoder has the same pseudo-random seed generator, which is not known to the source of the data sequence.

The scheme above can correspond to both universal and non-universal coding processes. In this thesis universal schemes are considered in which the probability estimate does not depend on assumption for the data model, and should provide good performance for a large set of possible models. Thus, it should have some mechanism that essentially "learns" the relevant parameters of the incoming data, and assign probabilities accordingly.

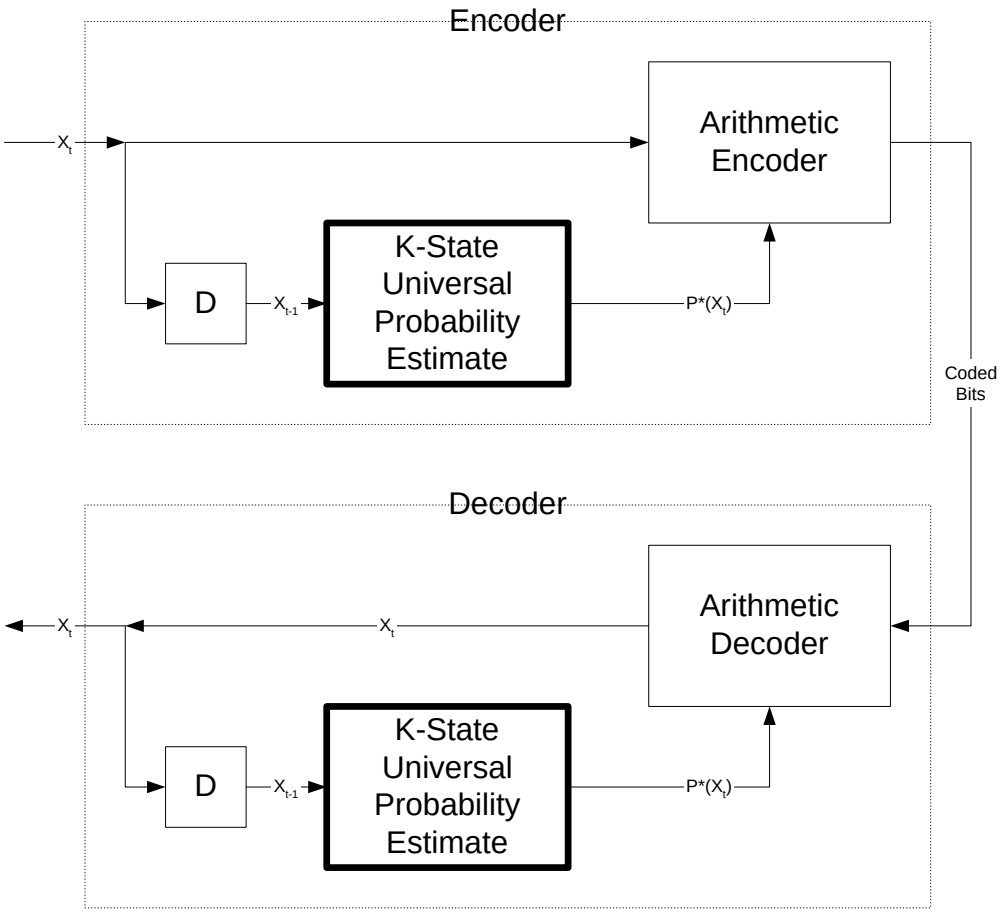

Figure 2.1: System architecture

2.2 Data Settings

Bernoulli setting

The simplest setting for universal coding is when data comes from an i.i.d. Bernoulli source, with unknown probability $p \triangleq Pr\{x_t = 1\}$. The reference compression rate is the source entropy, as defined in Equation (1.3):

$$\text{reference} \triangleq h(p). \tag{2.1}$$

Markovian setting

A more advanced stochastic setting is when data has Markov distribution of a known order q, with unknown values of the conditional probabilities $Pr(x_t = 1|x_{t-1}, \ldots, x_{t-q}) = p(x_t|s)$, at each Markov state s. The reference compression rate is the entropy of the source [13]:

$$\text{reference} \triangleq H_q = \sum_{s=1}^{2^q} P_s h(p(x_t|s)) \tag{2.2}$$

where s denotes the Markovian state, specified by the previous q symbols, and P_s is the stationary probability to be at state s.

Deterministic setting

In the deterministic setting, the data is an arbitrary infinite individual sequence. However, the performance of the universal encoder is compared to a reference value, which can be the empirical entropy of the sequence. This represents the best possible compression of a constrained batch encoder. The reference compression rate is the single-state empirical entropy:

$$\text{reference} \triangleq \rho_1(\underline{x}) = \limsup_{t\to\infty} h\left(\frac{N_t(1)}{t}\right). \tag{2.3}$$

The goal is to attain this rate for ANY sequence.

Higher order finite-state empirical entropy, or, finite-state compressibility, is also defined [15]:

$$\rho(\underline{x}) \triangleq \lim_{S\to\infty} \limsup_{t\to\infty} \min_{g\in G_s} \rho\left(g; x_1^t\right), \tag{2.4}$$

where

$$\rho\left(g; x_1^t\right) = \sum_{s=1}^{S} \frac{N_t(s)}{t} h\left(\frac{N_t(s,0)}{N_t(s)}\right). \tag{2.5}$$

This thesis does not consider this higher order deterministic reference.

2.3 Machine Types

The main constraint imposed on the universal machine analyzed here is that it has only K states. This work shall distinguish between three cases. The simplest case is where the machine is deterministic and time-invariant. A more flexible machine is a random machine, where the state transition function may be stochastic. This work shall also consider a time-variant machine, where the state transition function, and the probability assigned to each node, can vary in time.

Deterministic machine

A finite-state machine is a machine with a finite number of states, K. Following the processing of t data bits, $x_1, \ldots, x_t$, the machine remembers only S_t, the state at time t. S_0 is the initial state of the machine. The probability that the machine assigns for the bit x_{t+1} is depend only on S_t, meaning that each state has a fixed probability assigned to it, by the function:

$$\hat{p}_{t+1} = f\left(S_t\right). \tag{2.6}$$

The machine has a state-transition function, which is a function of the current state, and the next bit:

$$S_{t+1} = g\left(S_t, x_{t+1}\right). \tag{2.7}$$

Random machine

In a random machine, the initial state, S_0, and the state-transition function, $g(\cdot)$, are both "random" functions, i.e.,

$$S_0 = S_0(\Omega_0) \tag{2.8}$$

$$S_{t+1} = g\left(S_t, x_{t+1}, \Omega_{t+1}\right) \tag{2.9}$$

when Ω_i are independent random variables.

As shown in [6], random machines having the same K-state limitation as deterministic machine can provide better results. We shall see that same phenomena in this case as well.

Time-variant machine

In a time-variant machine, the probability function, $f(\cdot)$, and the state-transition function, $g(\cdot)$, are both time-variant functions, i.e.,

$$\hat{p}_{t+1} = f\left(S_t, t\right) \tag{2.10}$$

$$S_{t+1} = g\left(S_t, x_{t+1}, t\right). \tag{2.11}$$

2.4 Minimum Redundancy Goal

The self-information of an event x is the amount of information revealed by the fact that $x = 0$ or $x = 1$, given the a-priori probability p for this event

to occur:

$$i(x|p) \triangleq \left\{ \begin{array}{ll} -\log_2(1-p) & x=0 \\ -\log_2(p) & x=1 \end{array} \right\} \tag{2.12}$$

The actual compression rate, using idle code, of a given sequence is the sum of the self-information of all the events, divided by the length of the sequence:

$$\text{actual} \triangleq \limsup_{T\to\infty} \frac{\sum_{t=1}^{T} i(x_t|p_t)}{T} \tag{2.13}$$

and the redundancy is:

$$\text{redundancy} \triangleq \text{actual} - \text{reference}. \tag{2.14}$$

The goal is to minimize the redundancy, universally, by selecting the optimal machine, i.e., selecting S_0, $f(\cdot)$ and $g(\cdot)$.

Chapter 3

Optimal Probability Axis Quantization

This chapter analyzes an optimal quantization of the probability axis, where the goal is to minimize a worst case redundancy associated with the quantization process. This redundancy is an essential ingredient of any problem where a finite-state machine assigns probabilities. This is because a finite-state machine, with only K states, can assign at most K distinct values of probabilities p_t. Thus, the possible probability range $[0, 1]$ has to be "quantized" into K values.

Assuming data is coming from a Bernoulli source with (real) probability p, and this probability is estimated, after quantization, by some value p_i, where $i \in \{1, \ldots, K\}$. When encoding this data into coded bits, the encoder will have an inaccurate data model, in the case that $p \neq p_i$, causing it to be sub-optimal. Since p can have any value, and p_i can have only K values, this happens often. The average extra bit-rate over the entropy is the divergence between p and p_i, $\mathrm{D}\left(p \parallel p_i\right)$, as defined in Equation (1.4).

This provides a fundamental limitation on the achievable performance of

any finite-state machine. There is always a value of p for which the difference between p and the closest estimate p_i is at least $1/2K$.

3.1 Min-Max Criterion

One possible way to design an optimal quantization for probability assignment is to use the following min-max criterion:

$$[p_1, \ldots, p_K]^{opt} = \arg \min_{[p_1,\ldots,p_K]} \left(\max_p \left(\min_i \left(\mathrm{D}\left(p \parallel p_i\right)\right)\right)\right) \tag{3.1}$$

where the inner minimum is simply the selection of the closer (in divergence sense) point between the two closest quantization points of p from both sides. Denote in the sequel:

$$\mathrm{D}_{max} \triangleq \max_p \left(\min_i \left(\mathrm{D}\left(p \parallel p_i\right)\right)\right) \tag{3.2}$$

which is the maximal divergence given $[p_1, \ldots, p_K]$. The optimal quantization is defined as the one that minimizes D_{max}.

3.2 High Resolution Asymptotic Quantization

Theorem 3.1 *In the Bernoulli setting, for $K \gg 1$,*

$$\min_{[p_1,\ldots,p_K]} \mathrm{D}_{max} = \frac{1.78}{K^2} + \textit{higher order terms,} \tag{3.3}$$

and it is achieved by the quantization points:

$$p_i{}^{opt} = \sin^2\left(\frac{\pi(i-\frac{1}{2})}{2K}\right). \qquad ; i = 1, \ldots, K \tag{3.4}$$

Proof: At high resolution, the divergence function can be approximated as:

$$\mathrm{D}\left(p \parallel p \pm \epsilon\right) \approx \frac{\epsilon^2}{2\ln(2)p(1-p)} \qquad \left\{ \begin{array}{lcl} |\epsilon| & \ll & p \\ |\epsilon| & \ll & 1-p \end{array} \right\} \tag{3.5}$$

Define a density function, $\lambda(p)$, which is the density of the quantization points near point p,

$$\lambda(p) \geq 0 \quad \forall p, \qquad \int_0^1 \lambda(p)dp = K, \tag{3.6}$$

and we get

$$\epsilon_{max}(p) \approx \frac{1}{2\lambda(p)} \tag{3.7}$$

$$\mathrm{D}_{max}(p) \approx \frac{1}{8\ln(2)p(1-p)\lambda^2(p)} \tag{3.8}$$

Since min-max causes the $\mathrm{D}_{max}(p)$ to be constant for any point p:

$$\lambda(p) \propto (p(1-p))^{-1/2} \tag{3.9}$$

which gives:

$$\lambda(p) = \frac{K}{\pi\sqrt{p(1-p)}} \tag{3.10}$$

$$\mathrm{D}_{max}(p) \approx \frac{\pi^2}{8\ln(2)K^2} \approx \frac{1.78}{K^2} \tag{3.11}$$

The "restoration points", $\{p_i\}_1^K$, can be located by integration:

$$\int_0^{p_i} \lambda(p)dp = i - \frac{1}{2} \qquad ; i = 1, \ldots, K \tag{3.12}$$

which leads to Equation (3.4). ■

This is a lower bound on the performance of any finite-state encoder. To achieve this limit the encoder should "lock" on the right state at all times. As will be seen later, a time-variant machine can do that!

Graphical interpretation

It turns out that there is a graphical interpretation for these restoration points. The points are arranged on a trajectory of a linear-spaced circle over

the probability axis, as demonstrated in Figure 3.1. Note that the "decision points" of the quantizer are not the trajectory of the slices – they are slightly different.

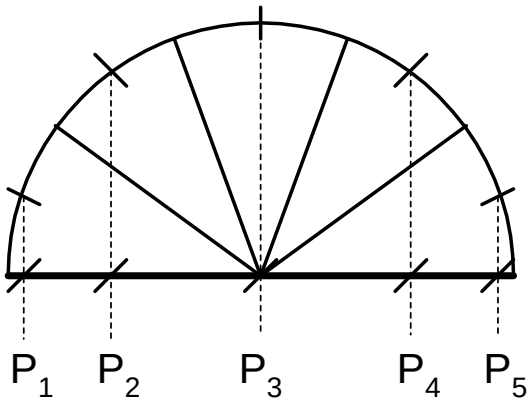

Figure 3.1: Optimal quantization points

Empirical results

When comparing this high-resolution quantization with the optimal one, for a given finite K, it seems that the differences are minor (even for reasonable K). For example, for $K = 100$, the distance between the optimal points and points given by Equation (3.4) is no more than 0.000192, which is only about 1% of the distance between the points. The optimal points were calculated using an algorithm which is similar to Lloyd scalar quantization algorithm [13]. A Java program created for comparing these results can be requested via e-mail to the author.

3.3 Comparison with Uniform Point Allocation

Under the divergence difference measure, the uniform point allocation performs poorly. The maximal divergence between a quantization point and a true probability point occurs at edges, i.e., when $p = 0$ or $p = 1$. The divergence at these points is:

$$\mathrm{D}_{max}(0) = -\log\left(1 - \frac{1}{2K}\right) \approx \frac{0.7213}{K} \tag{3.13}$$

which is $\Theta\left(\frac{1}{K}\right)$ instead of $\Theta\left(\frac{1}{K^2}\right)$ in the optimal quantization.

Under a Bayesian criterion, where it is assumed that p is uniform in the range $[0, 1]$, quantization with uniform point allocation does not perform well either. This is because the divergence at the edges is, as above, $\Theta\left(\frac{1}{K}\right)$. Approximating the difference for each point by $\epsilon = \frac{1}{2K}$, which is the worst case, leads to:

$$\mathrm{D}_{average} \approx 2\int_0^{\frac{1}{2}} \mathrm{D}\left(p \parallel p + \epsilon\right)dp = \frac{\ln(K)}{8\ln(2)K^2} \tag{3.14}$$

While this calculation is only approximate, we easily observed that for uniform distribution $\mathrm{D}_{average}$ behaves as:

$$\mathrm{D}_{average} = \Theta\left(\frac{\log K}{K^2}\right). \tag{3.15}$$

Chapter 4

Bernoulli Setting

This chapter considers the Bernoulli setting, and describes various finite-state machines that cope with it. It turns out that in this case there is an optimal redundancy rate for time-invariant machines of $\Theta\left(\frac{1}{K}\right)$. This work first introduces a random finite-state machine, and proves that it achieves this optimal convergence rate. Then, it provides a deterministic machine whose rate is $\Theta\left(\frac{\log K}{K}\right)$. This machine is constructed out of the random machine, by sacrificing few states, and using the true randomness of the Bernoulli data. Finally, this work provides a time-variant machine whose rate is $\frac{1.78}{(K-2)^2}$, i.e., it achieves the quantization limit itself(!), including the exact optimal constant.

4.1 Random Machine

Leighton and Rivest investigated an estimation problem using finite-state machines [11], related to our problem, but in a different context. In their paper, they used mean square error criterion, i.e., minimizing $(\hat{p}-p)^2$, and not the coding redundancy, i.e., minimum $\mathrm{D}\left(p \parallel \hat{p}\right)$. It is shown there that the best finite-state machine (either random or deterministic) cannot do better

than $\Theta\left(\frac{1}{K}\right)$, for every p. They also presented a random machine that attains this rate. This machine has uniform quantization structure.

In our case, same results are obtained by following their derivations, where instead of using a uniform quantization of the probability axis, we use the optimal quantization, given by Equation (3.4). This can be done because in high-resolution quantization the divergence function is actually a square error function, as showed by Equation (3.5), since the factor $p(1-p)$ becomes a constant. This gives a lower bound on the redundancy of $\Theta\left(\frac{1}{K}\right)$.

The suggested universal random machine with K states is given at Figure 4.1. This machine is essentially the optimal random machine suggested at [11], where the probability value at each node, and the transition probabilities, are set using the optimal quantization formula, given by Equation (3.4). These probabilities are denoted $r_1, \ldots, r_K$ in the figure, where $r_i + r_{K-i+1} = 1$. This machine attains the optimal $\Theta\left(\frac{1}{K}\right)$ error, for every p.

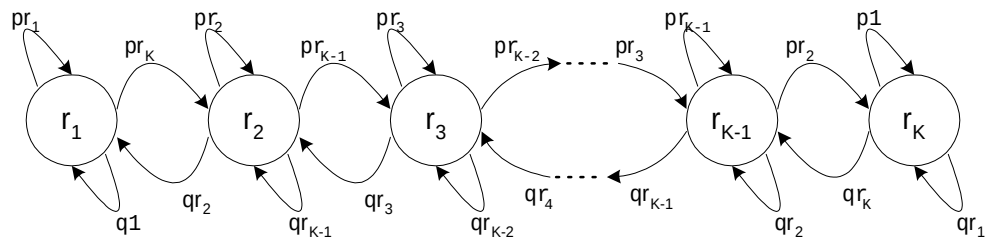

Figure 4.1: Random machine for the Bernoulli setting

Sliding-window interpretation

The machine above yields approximately the same results as a sliding-window deterministic machine, that remembers the last $K-1$ data bits explicitly, using 2^{K-1} states. The machine state, p_i, reflects that there are

$(K-1)\cdot p_i$ "set" bits in the buffer. Then, if the machine gets a set bit, $x_{t+1}=1$, it should remove x_{t-K+2} from the buffer, and append the set bit. Because the machine does not know the value of the removed bit, it assumes that it is one in probability p_i. Combined with the set bit we need to add, the machine remains in the same state in probability p_i, and advances one state in probability $1-p_i=p_{K-i+1}$.

Stable equilibrium interpretation

In order to continuously move towards the correct state, the machine should be in a stable equilibrium in that state. At any time t, the "pressure" on the machine is defined as the expectation of the state transition, given the current state:

$$pressure \triangleq E\{S_{t+1}-S_t \mid S_t\}. \tag{4.1}$$

By defining the estimation error at time t, $\epsilon_t \triangleq \hat{p}_t - p$, the pressure can be calculated:

$$\begin{aligned} pressure &= p\cdot(1-\hat{p}_t)-q\cdot\hat{p}_t \\ &= p\cdot(1-p-\epsilon_t)-(1-p)\cdot(p+\epsilon_t) \\ &= -\epsilon_t. \end{aligned} \tag{4.2}$$

When $\hat{p}_t = p$, the pressure is zero. When $\hat{p}_t \neq p$ the pressure on $\hat{p}_t$ is towards p, at the magnitude of the difference between them. Effectively, this gives a Gauss distribution for $\hat{p}$ where average is p and its variance is in the order of $\Theta\left(\frac{1}{K}\right)$.

4.2 Deterministic Machine

Following [11], the random machine above can be converted into deterministic machine, by using the true randomness of the Bernoulli data. This conversion expands each of the K states into $\Theta(\log K)$ states, thus reducing the convergence rate to $\Theta\left(\frac{\log K}{K}\right)$.

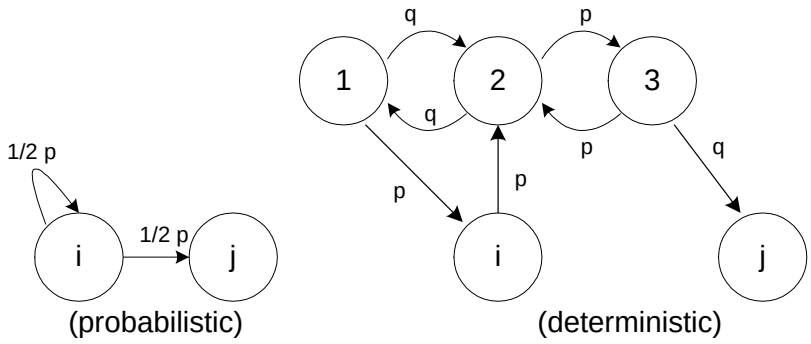

Figure 4.2: Simulation of probabilistic transitions by deterministic machine

The basic idea is to look at two data bits at a time, skipping all occurrences of 00, 11, and using only 01, 10. Since the data is Bernoulli, the probability to get 01 before 10, and vise versa, is the same (50%), independent of p. Figure 4.2 demonstrates this method, for $p_i = 1/2$. In order to implement it for, say, $p_i = 5/8$, we need 3 stages, or 9 intermediate states. In general, in order to get an accuracy of $1/Q$ we need $\Theta(\log Q)$ states.

In order to implement the random machine suggested in Figure 4.1, we need an accuracy based on Equation (3.4), of $Q \leq K^2$ at the edges of the probability axis. This means that each state is expanded into $\Theta(\log K)$ states, thus reducing the convergence rate to $\Theta\left(\frac{\log K}{K}\right)$, as described above.

4.3 Time-Variant Machine

One of the main results of this thesis is that the quantization limit, as derived in the previous chapter, is asymptotically achievable by a time-variant machine. This means that time-variant machine can have the same performance as time-invariant machine, but with a smaller, square root, number of states.

Theorem 4.1 *There exists a time-variant machine whose redundancy is,* $\forall p$*, at most*

$$\frac{1.78}{(K-2)^2} + \epsilon \qquad ; \forall \epsilon > 0 \tag{4.3}$$

Proof: Given K, set $K-2$ quantization points using the optimal quantization scheme, as defined in Equation (3.4). These "restoration points" will slice the $[0,1]$ range into $K-2$ intervals, each interval corresponds to all probability values associated with a specific quantization value. For every positive ϵ, the machine converges in a finite time and stays at the correct interval with probability $1-\epsilon$. The worse divergence, under the assumption that the machine is at the correct interval, was shown to be $\frac{1.78}{(K-2)^2}$. Since it converges in finite time to this interval, the average (over time) of the divergence will also not exceed this value.

To prove that the convergence occurs in finite time, one can utilize the results of Cover [3]. It was shown there that with only 4 states and infinite time, it is possible to recognize, with probability one, whether the machine is above or below a given decision point. This is done using iteration between 4 time-groups, as described in Figure 4.3. In order that this process will converge to the right point in probability one, there is a need to enlarge N_0 and N_1 towards infinity exactly at the rate specified in [3].

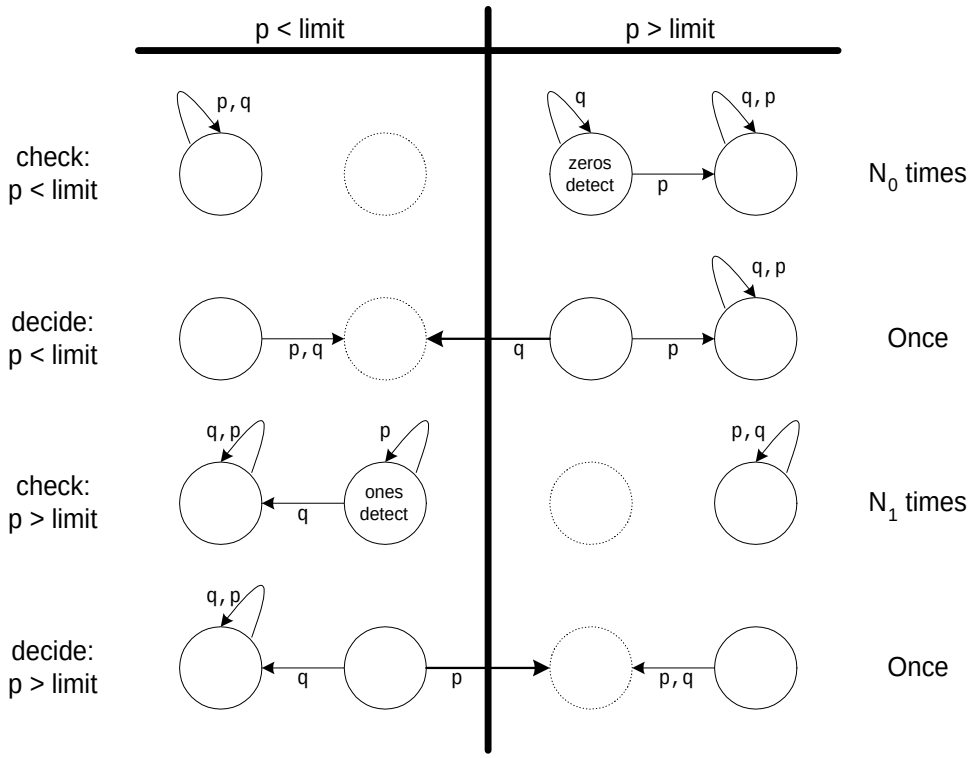

Figure 4.3: Cover's 4-state process

Let fix $\delta > 0$. It was also shown in [3] that there exists a finite time interval, $T(\delta, \alpha)$, so that with probability $1 - \delta$ the machine can determine within this time interval whether the source probability p is below or above α. In this case there are $K - 3$ such threshold points, each between two quantization points. Using this, it can be demonstrated how a time-variant machine can determine the correct interval with high enough probability, and within a finite duration (which is actually the sum of time durations require to check all these thresholds).

Specifically, the machine first checks whether p is below or above the first decision point d_1. After the finite time $\mathrm{T}(\delta, d_1)$, and using the 4 states, it makes the decision whether p is below or above d_1, and the probability of error in this decision is at most δ. If the machine decides that p is below d_1, it goes into a state that serves as a "sink" for the first quantization value. Otherwise it moves to the second stage.

At the n-th stage the machine checks whether p is below or above the n-th decision point d_n. Only $n - 1$ states are needed in order to remember if the machine entered a sink at any previous decision, and extra 4 states for making the threshold decision regarding d_n. This takes $\mathrm{T}(\delta, d_n)$ time.

At the last stage, where $n = K-3$, the machine needs $K-4$ states in order to remember previous sinks, and extra 4 states for the decision procedure, a total of K states. After $\mathrm{T}(\delta, d_{K-3})$ time it ends up at a sink, i.e., it is locked at a state, and ignores the rest of the data.

Each state has a probability of no more than δ to err. By setting $\delta = \frac{\epsilon}{K-3}$, the total probability of error will be no more than ϵ throughout this process.

■

Chapter 5

Markovian Setting (q-th order)

This chapter considers a stochastic setting where the data comes from an unknown q-th order Markovian process. The goal, of course, is to attain the Markovian entropy. To analyze this setting, the chapter first considers the problem of multi-dimensional quantization of the Markovian probability space, and provides a multi-dimensional quantization limit on the redundancy. With this quantization result, the machines suggested at the previous section, for the Bernoulli case, can be modified for the Markovian case. Specifically, the chapter ends with a presentation of a deterministic finite-state machine and a randomized finite-state machine that attain the q-th order entropy at a rate of $\Theta\left(\frac{\log K}{K^{2-q}}\right)$ and $\Theta\left(\frac{1}{K^{2-q}}\right)$ respectively.

5.1 Multi-Dimensional Quantization Limit

This section analyzes the multi-dimensional quantization of the Markovian probability space, and provides a multi-dimensional quantization limit on the redundancy.

To fully describe a q-th order Markov process, one needs 2^q independent

parameters (the probabilities of one at each state), whose values are in the range $[0,1]$. Assume that the parameters are bounded away from 0 and 1, considering only ergodic Markov processes. Thus, the process is associated with a single point inside a unit hyper-cube in the $\mathbb{R}^{2^q}$ Euclidean space. When any finite-state machine, with only K states, propagates according to a data sequence, it can only distinct between at most K sets of process parameters. The optimal thing to do, which is **not** achievable by any finite-state machine, is to divide the hyper-cube into K distinct regions. Each region will have a "restoration point" inside it.

Given that the "restoration point" is $\underline{a} \triangleq [a_1, a_2, \ldots, a_{2^q}]$, and the actual "process point" is $\underline{b} \triangleq [b_1, b_2, \ldots, b_{2^q}]$, the redundancy (average bit-rate over the entropy) is the multi-dimensional divergence from $\underline{b}$ to $\underline{a}$ which is given by:

$$\mathrm{D}\left(\underline{b} \parallel \underline{a}\right) \triangleq \sum_{s=1}^{2^q} f(\underline{b}, s) \mathrm{D}\left(b_s \parallel a_s\right) \tag{5.1}$$

where $f(\underline{b}, s)$ is the stationary probability that the source is in a Markovian state s, evaluated with the conditional probability vector $\underline{b}$ [13]. The divergence $\mathrm{D}\left(b_s \parallel a_s\right)$ is the scalar divergence between b_s and a_s, as defined in Equation (1.4).

Define the quantization limit, D_{qlimit}, as the lower limit on the multi-dimensional quantization redundancy over the entropy, using min-max criterion,

$$\mathrm{D}_{qlimit} \triangleq \min_{\{\underline{a}\}_1^K} \left(\max_{\underline{b}} \left(\mathrm{D}\left(\underline{b} \parallel \underline{a}\right) \right) \right). \tag{5.2}$$

Theorem 5.1 *In the q-th order Markovian setting, the quantization limit,* D_{qlimit}*, satisfies*

$$\mathrm{D}_{qlimit} = \Theta\left(\frac{1}{K^{2^{1-q}}}\right) \tag{5.3}$$

Proof: In order to evaluate D_{qlimit}, define $\delta_s \triangleq b_s - a_s$, $\Delta \triangleq \underline{b} - \underline{a}$, and define the hyper-cube B to be all the points (vectors) whose coefficients b_s satisfy, say, $0.4 \leq b_s \leq 0.6$. Now,

$$\begin{aligned}
\mathrm{D}_{qlimit} &= \min_{\{\underline{a}\}_1^K} \max_{\underline{b}} \mathrm{D}\left(\underline{b} \parallel \underline{a}\right) \\
&\geq \min_{\{\underline{a}\}_1^K} \max_{\underline{b}\in B} \mathrm{D}\left(\underline{b} \parallel \underline{a}\right) && (5.4) \\
&= \min_{\{\underline{a}\}_1^K} \max_{\underline{b}\in B} \sum_{s=1}^{2^q} f(\underline{b}, s)\mathrm{D}\left(b_s \parallel a_s\right) \\
&\geq \min_{\{\underline{a}\}_1^K} \max_{\underline{b}\in B} \sum_{s=1}^{2^q} 0.4^q \mathrm{D}\left(b_s \parallel a_s\right) && (5.5) \\
&\geq \min_{\{\underline{a}\}_1^K} \max_{\underline{b}\in B} \sum_{s=1}^{2^q} 0.4^q c_1 \delta_s^2 && (5.6) \\
&= c_1 0.4^q \min_{\{\underline{a}\}_1^K} \max_{\underline{b}\in B} \sum_{s=1}^{2^q} \delta_s^2 \\
&= c_2(q) \min_{\{\underline{a}\}_1^K} \max_{\underline{b}\in B} \|\Delta\|^2 && (5.7) \\
&\geq c_3(q) 2^q \left(\frac{0.6 - 0.4}{2m}\right)^2 && (5.8) \\
&= c_4(q)\frac{1}{m^2} && (5.9)
\end{aligned}$$

where the first inequality above (5.4) follows since $\underline{b}$ is maximized in a subset of the hyper-cube, the second inequality (5.5) comes from using a lower bound on $f(\underline{b}, s)$, the next inequality (5.6) follows from properties of the divergence, and the equality immediately after it (5.7) just follows from the definition of the Euclidean norm.

Define by m the number of point at each axis, the last two equations follow (5.8) by expressing the volume of the optimal grid given by the setting $\{\underline{a}\}_1^K$ in terms of the uniform grid, which has m values in each dimension, and so requires $K = m^{2^q}$ states. Combining all these relations yields $\mathrm{D}_{qlimit} > c_4(q)\frac{1}{m^2}$, which leads, after a short calculation, to Equation (5.3). ■

Note that the optimal point allocation is **not** independent in each dimension, as showed before in vector quantizers [10]. However, the gain with optimal packing over the hyper-cube arrangement above will be only a constant, and will not affect the asymptotic behavior of the redundancy as a function of K.

5.2 Deterministic Machine

The deterministic machine for attaining the entropy of the q-th order Markov source, is essentially composed of 2^q machines, each of the form of the deterministic machine for the Bernoulli case, running simultaneously. Each of the states in this machine actually correspond to 2^q states, since the q-th order Markovian history of that quantization state should be remembered. Thus the machine has a total of $\Theta\left(2^q(m\log m)^{2^q}\right)$ states in order to get a redundancy of $\Theta\left(\frac{1}{m}\right)$, which gives a redundancy of $\Theta\left(\frac{\log K}{K^{2^{-q}}}\right)$.

As in the Bernoulli case, this deterministic machine is worse (by a factor of $\log K$) than the square root of the quantization limit. In the Bernoulli case, we showed that the square root of the quantization limit is the optimal rate. Here, we can only conjecture that $\Theta\left(\frac{1}{K^{2^{-q}}}\right)$ is the real lower limit on this Markovian time-invariant machine, because of the Gauss distribution of the "restoration point", $\underline{a}$, around the actual "process point", $\underline{b}$. If this is correct, this machine is only far by a factor of $\Theta(\log K)$ than the theoretical limit.

5.3 Random Machine

The proposed randomized machine in this case is again essentially composed of 2^q machines, each of the form of the randomized machine for the Bernoulli case (Figure 4.1), running simultaneously. Again, each of the states in this machine corresponds to 2^q states, since the machine needs to remember the q-th order Markovian history of that quantization state. Thus the machine has a total of $\Theta\left(2^q m^{2^q}\right)$ states in order to get a redundancy of $\Theta\left(\frac{1}{m}\right)$, which gives a redundancy of $\Theta\left(\frac{1}{K^{2-q}}\right)$.

As in the Bernoulli case, this randomized machine attains the square root of the quantization limit. From the same reasons as in the case of the deterministic machine above, we conjecture that the real lower limit of the redundancy is in order of $\Theta\left(\frac{1}{K^{2-q}}\right)$, and this machine attains it.

Chapter 6

Deterministic Setting (single-state reference)

This chapter considers the deterministic setting, where the data is arbitrary, and the goal of the universal encoder is to attain the single-state empirical entropy of the data. The chapter starts by presenting a general concept of minimal circles, and then, apply this concept in order to understand how deterministic machine processes deterministic data. While the optimal performance (convergence rate of the redundancy) for finite-state universal deterministic machines were not determined, a deterministic machine with good redundancy performance is described. The chapter ends with preliminary examination of random finite-state universal machines, and time-variant machines for this deterministic setting.

6.1 Minimal Circles

Consider the case where the data is deterministic, and the finite-state machine is deterministic and time-invariant. We now show a constructive

method to find the worst sequence of data for a given machine, i.e., the one that has the highest redundancy over its single-state empirical entropy.

Define a **circle** as a cyclical, L-elements, ordered set of states, $\{S_i\}_0^{L-1}$, and input bits, $\{x_i\}_1^L$, that rotate the machine between the states, i.e., satisfies:

$$\begin{aligned} g\left(S_0, x_1\right) &= S_1 \\ g\left(S_1, x_2\right) &= S_2 \\ &\vdots \\ g\left(S_{L-2}, x_{L-1}\right) &= S_{L-1} \\ g\left(S_{L-1}, x_L\right) &= S_0. \end{aligned} \tag{6.1}$$

Calculating the reference compressed bit-rate, the actual bit-rate, and the redundancy of a given circle, $\underline{c}$, gives these results:

$$\text{reference}(\underline{c}) = h\left(\frac{\sum_{i=1}^{L} x_i}{L}\right) \tag{6.2}$$

$$\text{actual}(\underline{c}) = \frac{1}{L}\sum_{i=1}^{L} i(x_i|p_i) \tag{6.3}$$

$$\text{redundancy}(\underline{c}) = \text{actual}(\underline{c}) - \text{reference}(\underline{c}). \tag{6.4}$$

Note that the redundancy in a given circle can be zero, positive, or negative. Clearly, in order not to violate the quantization limit, there has to be at least one circle with a positive redundancy.

Define a **minimal circle** as a circle that does not contain the same state twice, meaning,

$$S_i \neq S_j \qquad \forall i \neq j; \quad i, j \in \{1, \ldots, L\}. \tag{6.5}$$

Lemma 6.1 *The number of different minimal circles is finite.*

Proof: Since every minimal circle cannot have the same state twice, their maximal length is K hops. Each hop in the circle can have at most K values for S_i and two values for x_i, giving no more than $(2K)^K$ different minimal circles (actually, there is much less). ■

Lemma 6.2 *Any non-minimal circle, $\underline{c}$, can be broken into an ordered set of minimal circles, $\{\underline{c}_1, \underline{c}_2, \ldots, \underline{c}_m\}$.*

Proof: A constructive method to achieve it is given here. First, search for the first occurrence of a repeating state, i.e., find the minimal i that satisfies

$$S_i = S_j \qquad \exists j \in \{0, 1, \ldots, i-1\} \tag{6.6}$$

and take out the states $S_j, \ldots, S_{i-1}$, with their input bits $x_{j+1}, \ldots, x_i$, and move them from $\underline{c}$ to $\underline{c}_1$. Then, repeat this process to define $\{\underline{c}_2, \underline{c}_3, \ldots, \underline{c}_{m-1}\}$. Finally, the rest of the states are $\underline{c}_m$ ■

For example, the circle in Figure 6.1 can be broken into 3 minimal circles:

$$\begin{aligned} \underline{c} &= 1, 2, 3, 4, 5, 6, 3, 2, 7, 8 \\ &\Downarrow \\ \underline{c}_1 &= 3, 4, 5, 6 \\ \underline{c}_2 &= 2, 3 \\ \underline{c}_3 &= 1, 2, 7, 8 \end{aligned}$$

Lemma 6.3 *The redundancy of the whole circle, $\underline{c}$, is less than or equal to the weighted average of the redundancies of the minimal circles, $\{\underline{c}_1, \underline{c}_2, \ldots, \underline{c}_m\}$. Equality happens only when each minimal circle has the same empirical probability.*

$$\underline{c} = \{\underline{c}_1, \underline{c}_2, \ldots, \underline{c}_m\} \tag{6.7}$$

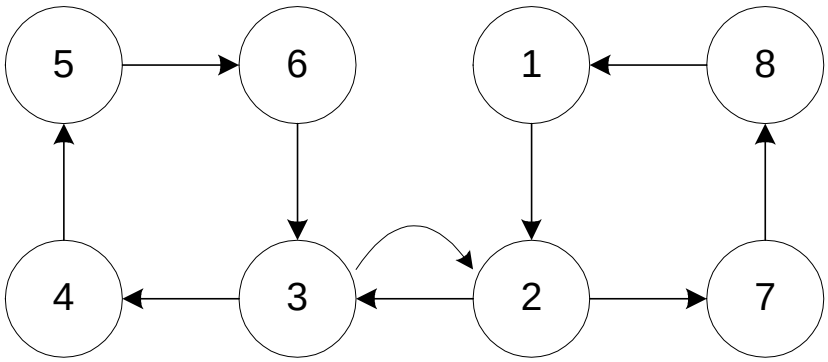

Figure 6.1: Minimal circles

$$\Downarrow$$

$$redundancy(\underline{c}) \leq \frac{1}{L}\sum_{i=1}^{m} L_i \cdot redundancy(\underline{c}_i) \tag{6.8}$$

Proof: From the linearity of Equation (6.3), we note that

$$\text{actual}(\underline{c}) = \frac{1}{L}\sum_{i=1}^{m} L_i \cdot \text{actual}(\underline{c}_i) \tag{6.9}$$

and from Equation (6.2), we note that

$$\begin{aligned}
\text{reference}(\underline{c}) &= h\left(\frac{\sum_{i=1}^{L} x_i}{L}\right) \\
&= h\left[\frac{1}{L}\left(L_1\frac{\sum_{i=1}^{L_1} x_{1,i}}{L_1} + \cdots + L_m\frac{\sum_{i=1}^{L_m} x_{m,i}}{L_m}\right)\right] \\
&\geq \frac{1}{L}\left[L_1 h\left(\frac{\sum_{i=1}^{L_1} x_{1,i}}{L_1}\right) + \cdots + L_m h\left(\frac{\sum_{i=1}^{L_m} x_{m,i}}{L_m}\right)\right] \qquad (6.10) \\
&= \frac{1}{L}\left[L_1\text{reference}(\underline{c}_1) + \cdots + L_m\text{reference}(\underline{c}_m)\right] \\
&= \frac{1}{L}\sum_{i=1}^{m} L_i \cdot \text{reference}(\underline{c}_i) \qquad (6.11)
\end{aligned}$$

when inequality (6.10) is because $h(\cdot)$, the binary entropy function define in Equation (1.3), is concave, and using Jensen inequality. At this point, the

redundancy can be calculated,

$$\begin{aligned}
\text{redundancy}(\underline{c}) &= \text{actual}(\underline{c}) - \text{reference}(\underline{c}) \\
&\leq \frac{1}{L}\sum_{i=1}^{m} L_i \cdot \text{actual}(\underline{c}_i) - \frac{1}{L}\sum_{i=1}^{m} L_i \cdot \text{reference}(\underline{c}_i) \\
&= \frac{1}{L}\sum_{i=1}^{m} L_i \cdot \text{redundancy}(\underline{c}_i) \qquad (6.12)
\end{aligned}$$

which proves Equation (6.8). ■

Lemma 6.4 *There exists a minimal circle with redundancy at least the redundancy of the whole circle, i.e.,*

$$\begin{aligned}
\exists k &\in \{1, 2, \ldots, m\} \\
\textit{redundancy}(\underline{c}_k) &\geq \textit{redundancy}(\underline{c}) \qquad (6.13)
\end{aligned}$$

Proof: Assume that it is false, i.e.,

$$\begin{aligned}
\forall k &\in \{1, 2, \ldots, m\} \\
\text{redundancy}(\underline{c}_k) &< \text{redundancy}(\underline{c}) \qquad (6.14)
\end{aligned}$$

then, use Equation (6.8)

$$\begin{aligned}
\text{redundancy}(\underline{c}) &\leq \frac{1}{L}\sum_{i=1}^{m} L_i \cdot \text{redundancy}(\underline{c}_i) \\
&< \frac{1}{L}\sum_{i=1}^{m} L_i \cdot \text{redundancy}(\underline{c}) \\
&= \text{redundancy}(\underline{c}) \qquad (6.15)
\end{aligned}$$

which is a clear contradiction of the assumption. ■

Theorem 6.5 *The worst data sequence for a given deterministic, time-invariant machine, is the one that take the machine into the minimal circle with the highest redundancy, and rotate in it endlessly.*

Proof: Combining all the above gives a constructive method for finding the worst data sequence. First, find all minimal circles, which is a finite set, according to Lemma 6.1. Then, calculate the redundancy in each of them, using Equation (6.4). According to Lemma 6.4, there is no other non-minimal circle that has higher redundancy. Finally, each infinite sequence of data is a small set of transient states (less than K), that can be ignored, and a huge circle in the middle. ■

6.2 Deterministic Machine

In the deterministic setting, this work did not found lower bounds that will determine the optimal performance. However, after analyzing the performance of some finite-state machines in this setting, few interesting observations came up. It turns out that the deterministic machine suggested above, for the stochastic Bernoulli case, does not perform well in the deterministic setting. This happens because this machine has a lot of small, unbalanced minimal circles.

From the previous section, we conjecture that in order to have a low-redundancy machine, the machine need to have balanced circles. Since small circles seems to be unbalanced, an appealing configuration would be where all the circles have the same length, and all are balanced, e.g., have the same redundancy.

The best deterministic machine we have found for this setting is described in Figure 6.2, and performs as well as $\Theta\left(\frac{\log K}{\sqrt{K}}\right)$ for all sequences. It works as follows: up to time r it counts the number of zeros and ones, and assigns probabilities that the next bit is one using the Krichevsky-Trofimov estimate, as defined in Equation (1.7). At time r the machine resets.

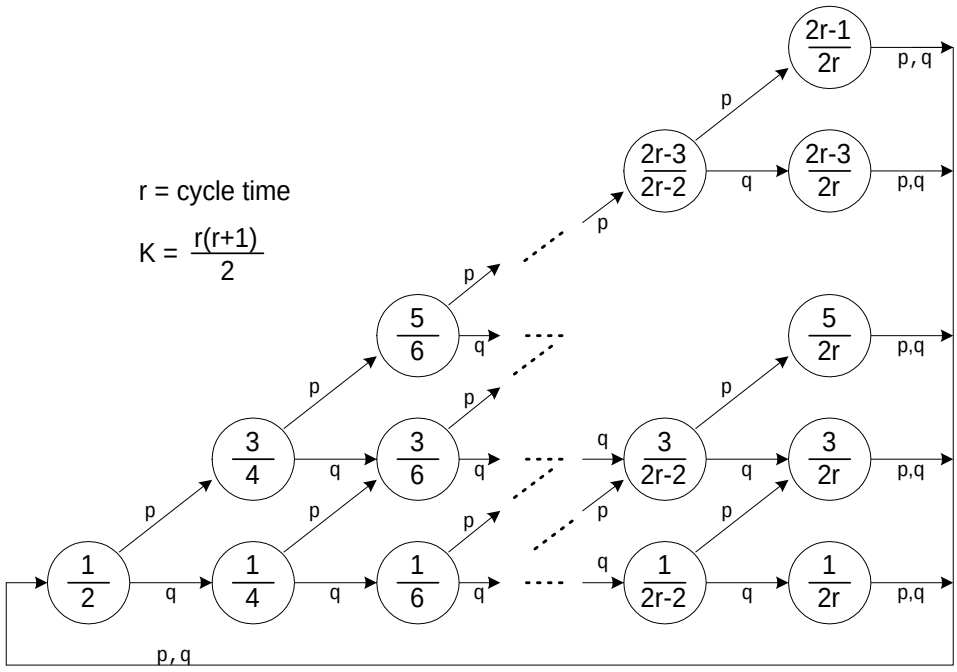

Figure 6.2: Deterministic machine for the deterministic setting

When analyzing this machine using the minimal circle theorem, it is easy to see that it is well balanced. This machine has 2^r minimal circles, when each of the circles has the same length, $L = r$, and the same redundancy, $\frac{1}{2r} \cdot \log_2(r+1)$. Since this machine has $K = \frac{r(r+1)}{2}$ states, the redundancy is $\Theta\left(\frac{\log K}{\sqrt{K}}\right)$.

Why full reset?

Although a full reset does not seem intuitively appealing, this is the best machine we have found that competes with all sequences. The reset maximizes the length of the minimal circles, thus lowers the redundancy. While remembering some state lowers the average redundancy, it creates some bad data sequences, which degrades performance measured using a min-max criterion.

Limited memory is sometimes beneficial!

Limited memory can be an advantage when the data is not stationary. For example, a long run of zeros, followed by a long run of ones, might be compressed better by the machine in Figure 6.2 when K is small. This happens because limited memory causes the machine to "forget" faster, thus lowering the redundancy caused by the non-stationary shifts in the data. When K is too small, however, the redundancy will tend to go up again, since the effect of the quantization become significant. This phenomena has been observed for prediction with limited memory in [17, 18].

Is there a lower limit on minimal circles redundancy?

It seems that there should be a lower limit on the redundancy of the minimal circles. For example, if one minimal circle has a negative redundancy, other should have a higher positive one. Also, small circles seems to be unbalanced. It seems that there should be some inequality that provides constrains on the redundancy of the minimal circles, other than the quantization limit found above.

After finding such a limitation, we think that the lower bound on the performance of any deterministic machine will be $\Theta\left(\frac{1}{\sqrt{K}}\right)$ for all sequences, and the machine in Figure 6.2 is only far by a factor of $\Theta\left(\log K\right)$ from it.

6.3 Random Machine

A possible randomized machine that can attain the empirical entropy, for any individual sequence, at an optimal rate $\Theta\left(\frac{1}{K}\right)$ of the redundancy, is the randomized machine suggested in Figure 4.1 for the Bernoulli case.

While we cannot prove that, the following reasoning suggests that this is a reasonable conjecture. Suppose the sequence exhibits a stationary behavior, i.e., its empirical distribution is about the same at different portion of the sequence. In this case, this machine would behave like in a case where the data comes from a stochastic Bernoulli source. In other words, the redundancy behaves as $\Theta\left(\frac{1}{K}\right)$.

On the other hand, if the sequence shows a non-stationary behavior (meaning that the empirical probability at various portions of the sequence vary, above and below the empirical probability of the entire sequence) it seems that the compression performance of this machine is only getting better. To see that, suppose that the instantaneous empirical probability is

changed from p_1 to p_2 and vice-versa, over a period of N bits each time. The machine has the property that as the empirical probability of the data changes, the probability estimate of the machine tracks it in a non uniform way – it initially changes with large steps and then smaller steps. As long as the machine estimate did not reach the middle point, it compresses poorer than the reference, but once it passes it, it compresses better. But because of the non-uniform tracking, a better compression than the reference is exhibited at a larger portion of the time.

The proof of this conjecture is left for further work.

6.4 Time-Variant Machine

The best time-variant machine we found performs as well as $\Theta\left(\frac{\log K}{K}\right)$ for all sequences. This machine, described in Figure 6.3, is similar to the deterministic time-invariant machine, but with one important exception: it does not need to allocate states in order to know t. This machine actually gives the exact estimations as the deterministic machine defined above with $r = K$. Using the same calculation, the redundancy of this machine is $\Theta\left(\frac{\log K}{K}\right)$.

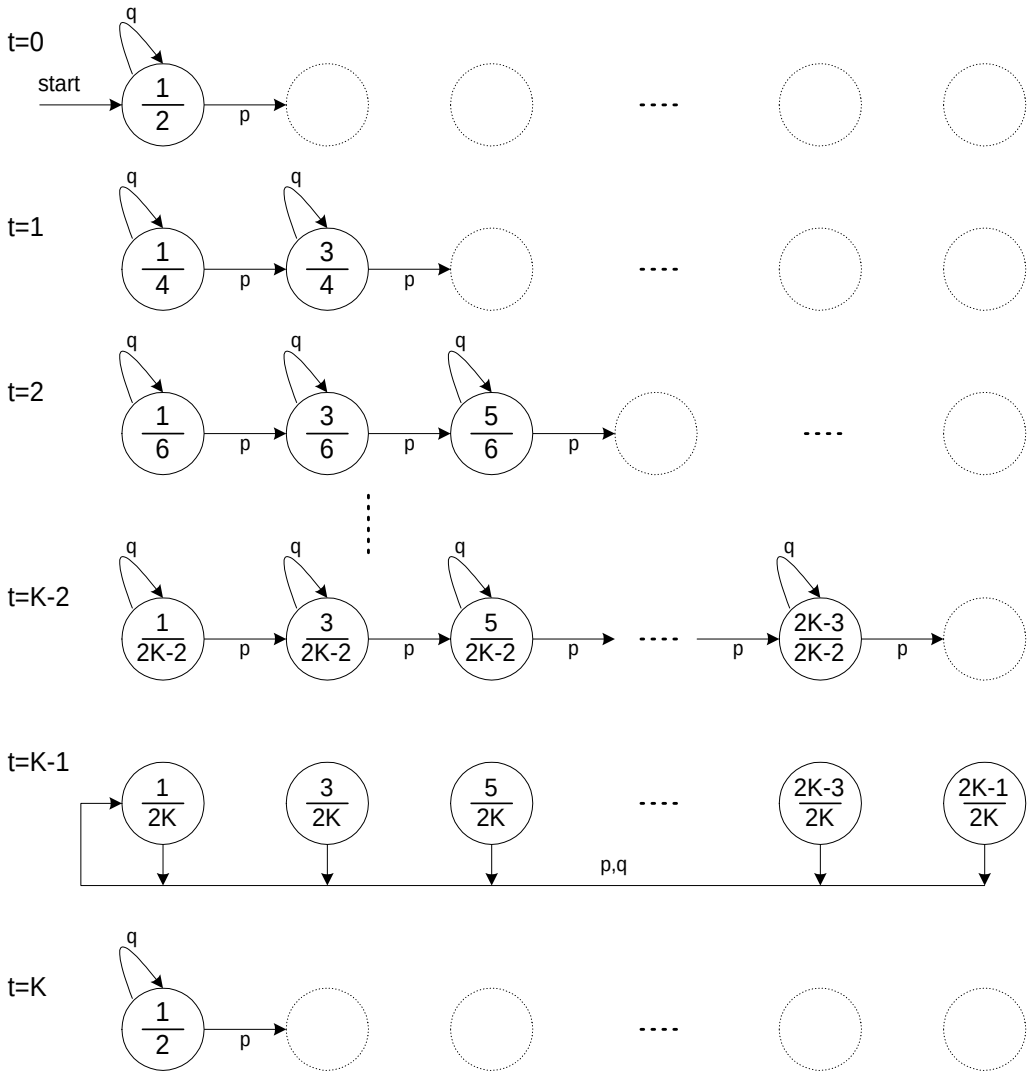

Figure 6.3: Time-variant machine for the deterministic setting

Chapter 7

Conclusion and Further Work

Before this work, it was well known that there exist universal encoders for binary sequences that can compete, for any individual sequence, with the best finite-state encoder tuned to that sequence. Similarly, universal encoders that attain the entropy of finite-state stochastic model were well known. However, these universal encoders required an infinitely growing number of states. This thesis addressed what happens when restricting the universal encoder itself to have a finite number of states.

To simplify the problem, the thesis considered universal finite-state machines that assign probabilities to the next bit. It ignored the additional number of (finite) states needed to translate the assigned probabilities into code bits.

The main results obtained are lower bounds on the redundancy as a function of the number of states, for the stochastic setting, in the Bernoulli and the q-th order Markov cases. Since the assigned probabilities have to be quantized (each value is associated with a state) these bounds come from examining the best possible quantization of the probability axis. Interestingly, in the Bernoulli case, the optimal quantization bounds can be attained

by time-varying machines. The work further develops bounds and provide specific machines for several additional cases, i.e., for time-invariant (random and deterministic machines) in the Bernoulli and the q-th order Markov setting.

The thesis also provides an initial examination for the deterministic setting. It describes universal finite-state machines with good performance, and even conjecture their optimality.

Following the thesis, there are several open problems and issues that are left for further work. In general, in several of the cases considered here the lower bounds on the redundancy are far from the performance attained by the suggested machines. In most these cases, we conjecture that the suggested machines represent the optimal performance.

Specific open problem that were mentioned along the thesis are:

- Is the upper limit found for the Markovian setting, using time-invariant machine tight? (see section 5.2 on page 40).
- Find a lower bound on the redundancy of the minimal circles (see page 50).
- Can we prove that the random machine for Bernoulli sequences is robust, i.e., can it work in the deterministic case (see section 6.3 on page 50).
- The problem of competing with higher order finite-state empirical entropy (the finite-state compressibility) in the deterministic setting is completely open (see Equation (2.4) on page 22).

While the general problem of universal coding with limited resources is still open, the thesis provides a set of basic results, such as the optimal quantization of the probability axis, and the proposed structures of the universal

machines, that will be useful in analyzing the general problem. These results thus provide an important step in understanding the problem of limited resources universal coding.

Bibliography

[1] C. E. Shannon, "A Mathematical Theory of Communication", Bell Syst. Tech. J., vol. 27, pp. 379–423, July 1948.

[2] A. N. Kolmogorov, "Three approaches to the quantitative definition of information", Probl. Inform. Transm., vol. 1, pp. 4–7, 1965.

[3] T. M. Cover, "Hypothesis testing with finite statistics", The annals of Mathematical Statistics, Vol. 40, No. 3, 828–835, 1969.

[4] M. E. Hellman, T. M. Cover, "Learning with finite memory", The annals of Mathematical Statistics, Vol. 41, No. 3, 765–782, 1970.

[5] T. M. Cover, M. E. Hellman, "The Two-Armed-Bandit Problem With Time-Invariant Finite Memory", IEEE Transactions on Information Theory, VOL. IT-16, No. 2, March 1970.

[6] M. E. Hellman, T. M. Cover, "On memory saved by randomization", The annals of Mathematical Statistics, Vol. 42, No. 3, 1075–1078, 1971.

[7] J. Ziv, A. Lempel, "A universal algorithm for sequential data compression", IEEE Transactions on Information Theory, VOL. IT-23, pp. 337–343, July 1977.

[8] J. Ziv, A. Lempel, "Compression of individual sequences via variable rate coding", IEEE Transactions on Information Theory, VOL. IT-24, pp. 530–536, September 1978.

[9] R. E. Krichevsky, V. K. Trofimov, "The Performance of Universal Encoding", IEEE Transactions on Information Theory, VOL. IT-27, No. 2, pp. 199-207, March 1981.

[10] R. M. Gray, "Vector quantization", IEEE-ASSP Magazine, pp. 4–29, April 1984.

[11] F. T. Leighton, R. L. Rivest, "Estimating a Probability Using Finite Memory", IEEE Transactions on Information Theory, VOL. IT-32, No. 6, November 1986.

[12] W. B. Pennebaker, J. L. Mitchell, "Probability estimation for the Q-coder", IBM Jour. of Res. and Dev., 32(6), pp. 737–751, November 1988.

[13] T. M. Cover, J. A. Thomas, "Elements of Information Theory", Wiley, NY, 1991.

[14] M. Feder, "Gambling Using a Finite State Machine", IEEE Transactions on Information Theory, VOL. IT-37, No. 5, September 1991.

[15] M. Feder, N. Merhav, M. Gutman, "Universal Prediction of Individual Sequences", IEEE Transactions on Information Theory, VOL. IT-38, No. 4, July 1992.

[16] N. Merhav, M. Feder, "Universal Prediction", IEEE Transactions on Information Theory, VOL. IT-44, No. 6, October 1998.

[17] M. Feder, E. Federovski, "Prediction of Binary Sequences Using Finite Memory", Proceedings of the 1998 International Symposium on Information Theory, Cambridge, MA, USA, August 1998.

[18] E. Federovski, "Universal Prediction of Binary Sequences Using Finite Memory", M.Sc thesis, Dept. of EE-Systems, Tel-Aviv University, 1998.

[19] D. Rajwan, M. Feder, "Universal Finite Memory Machines for Coding Binary Sequences", Proceedings of the 2000 Data Compression Conference, Snowbird, Utah, USA, pp. 113–122, March 2000.