Entropy reduction, ordering in sequence spaces, and semigroups of non-negative matrices

Henk D.L. Hollmann * Peter Vanroose †

October 18, 1995

Abstract

We develop a mathematical framework to investigate classification or ordering of sequential input by means of finite-state algorithms, where the aim is to reduce the "diversity" at the output, that is, to achieve entropy reduction.

Our main interest is in optimal time-varying strategies; here, given a (finite) collection of algorithms sharing a common set of internal states, we consider ordering strategies represented by sequences of these algorithms, where the action on the t-th input is determined by the t-th algorithm in the sequence. So, in a sense we consider a programmable finite-state device and we are looking for the best program.

Surprisingly, there is a uniform method to handle questions of this type. Indeed, we first show how to transform such a problem to a problem on eigenvalues in a related semigroup of non-negative matrices, and then we present an approach to this eigenvalue problem which seems to succeed most of the time.

We apply our methods to a problem concerning ordering in sequence spaces introduced by Ahlswede, Ye, and Zhang (see the references) which motivated part of this work. In particular, we show that $\tau_2(0,2,1) = \frac{1}{3}{}^2\log(2+\sqrt{3})$, as conjectured by the second author some years ago.

Keywords: Entropy reduction, classification, ordering, sofic system, finite state machine, Perron-Frobenius, semigroups of non-negative matrices.

*Philips Research Laboratories, Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands, hollmann@prl.philips.nl.

†Katholieke Universiteit Leuven, Kard. Mercierlaan 94, B–3001 Heverlee, Belgium, vanroose@esat.kuleuven.ac.be.

1 Introduction

In this paper, a *(time-invariant) finite-state entropy-reduction algorithm*, or briefly, an *algorithm*, is a synchronous finite-state input/output device; such a device takes inputs from a given source alphabet $\mathbf{B}$, and, depending on the input symbol and its internal state, produces an output symbol in an output alphabet $\mathbf{F}$ and moves to a new internal state.

In the context of channel codes (modulation codes), such a device is usually referred to as a (synchronous) finite-state encoder [6] and is used to translate or encode arbitrary sequences of source symbols into sequences that have certain desirable properties. In that context, it is of course required that decoding is possible.

Here, we think of such devices as performing some sort of data compression, and we will be interested in algorithms that have a "small" output space. A natural measure of the efficiency of an algorithm is thus the *entropy* of the output space, which measures the growth rate of the number N_n of output sequences of length n. (In the context of channel codes, the entropy is usually called the *(Shannon) capacity*.) We allow the case where the input sequences are restricted to a given constrained system over $\mathbf{B}$.

The use of the term "data compression" may cause confusion, since we do not consider the reconstruction problem. Instead, it might be better to speak of *ordering*, *classification*, or *entropy-reduction* of data sequences: an algorithm assigns an *index* to each data sequence (namely the corresponding output sequence) and the efficiency of the entropy-reduction is measured by the number of distinct indices produced by the algorithm.

As is well-known, the efficiency of such an algorithm, that is, the capacity of the constrained system formed by the collection of output sequences that can be produced by the algorithm, can be computed as the largest eigenvalue of a certain non-negative matrix.

Now let $\mathcal{F}$ be a finite collection of algorithms, all having the same source alphabet and sharing a common set of internal states. A *time-varying (entropy-reduction) algorithm over* $\mathcal{F}$ is a sequence

$$\mathbf{f} = f_1, f_2, f_3, \cdots$$

of algorithms $f_i \in \mathcal{F}$. We think of such a sequence $\mathbf{f}$ as an algorithm whose action at time t is directed by algorithm f_t. So at time t, $t = 1, 2, \ldots$, the algorithm $\mathbf{f}$ takes an input from the source alphabet and, depending on this input and its internal state, produces an output and moves to another internal state according to algorithm f_t. The collection of all entropy-reduction algorithms over $\mathcal{F}$ (all sequences over $\mathcal{F}$) will be denoted by $\mathcal{F}^\infty$.

It may happen that some time-varying algorithm in $\mathcal{F}^\infty$ performs better than the best algorithm in $\mathcal{F}$. (We will see examples of this later.) So the question now arises how to produce *lower bounds* for the efficiency of algorithms in $\mathcal{F}^\infty$ and how to find the best time-varying algorithm in $\mathcal{F}^\infty$. We will refer to this problem as the *optimal*