Pattern Recognition Letters 112 (2018) 49–55

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Extended-alphabet finite-context models

João M. Carvalho[a,*], Susana Brás[a,b], Diogo Pratas[a], Jacqueline Ferreira[c,d], Sandra C. Soares[c,e,f], Armando J. Pinho[a,b]

[a] Institute of Electronics and Informatics Engineering of Aveiro, University of Aveiro, Portugal
[b] Department of Electronics Telecommunications and Informatics of Aveiro, University of Aveiro, Portugal
[c] Department of Education and Psychology, University of Aveiro, Portugal
[d] IBILI, Faculty of Medicine, University of Coimbra, Portugal
[e] CINTESIS-UA, University of Aveiro, Portugal
[f] Department of Clinical Neurosciences, Karolinska Institute, Stockholm, Sweden

ARTICLE INFO

Article history:
Received 29 June 2017
Available online 1 June 2018

ABSTRACT

The normalized relative compression (NRC) is a recent dissimilarity measure, related to the Kolmogorov complexity. It has been successfully used in different applications, like DNA sequences, images or even ECG (electrocardiographic) signal. It uses a compressor that compresses a target string using exclusively the information contained in a reference string. One possible approach is to use finite-context models (FCMs) to represent the strings. A finite-context model calculates the probability distribution of the next symbol, given the previous k symbols. In this paper, we introduce a generalization of the FCMs, called extended-alphabet finite-context models (xaFCM), that calculates the probability of occurrence of the next d symbols, given the previous k symbols. We perform experiments on two different sample applications using the xaFCMs and the NRC measure: ECG biometric identification, using a publicly available database; estimation of the similarity between DNA sequences of two different, but related, species – chromosome by chromosome. In both applications, we compare the results against those obtained by the FCMs. The results show that the xaFCMs use less memory and computational time to achieve the same or, in some cases, even more accurate results.

1. Introduction

Data compression models have been used to address several data mining and machine learning problems, usually by means of a formalization in terms of the information content of a string or of the information distance between strings [1–5]. This approach relies on solid foundations of the concept of algorithmic entropy and, because of its non-computability, approximations provided by data compression algorithms [6].

A finite-context model (FCM) calculates the probability distribution of the next symbol, given the previous k symbols. In this work, we propose an extension of the FCMs, which we call extended-alphabet finite-context models (xaFCM). Usually, these models provide better compression ratios, leading to better results for some applications, especially when using small alphabet sizes – and also by performing much less computations. We show this in practice for the ECG biometric identification and DNA sequence similarity. The source code for the compressor was implemented using Python 3.5 and is publicly available under the GPL v3 license.[1]

1.1. Compression-based measures

Compression-based distances are tightly related to the Kolmogorov notion of complexity, also known as algorithmic entropy. Let x denote a binary string of finite length. Its **Kolmogorov complexity**, $K(x)$, is the length of the shortest binary program x^* that computes x in a universal Turing machine and halts. Therefore, $K(x) = |x^*|$, the length of x^*, represents the minimum number of bits from which x can be computationally retrieved [7].

The **Information Distance** (ID) and its normalized version, the **Normalized Information Distance** (NID), were proposed by Bennett *et al.* almost two decades ago [8] and are defined in terms of the Kolmogorov complexity of the strings involved, as well as the complexity of one when the other is provided.

However, since the Kolmogorov complexity of a string is not computable, an approximation (upper bound) for it can be used by

* Corresponding author.
E-mail address: joao.carvalho@ua.pt (J.M. Carvalho).

[1] https://github.com/joaomrcarvalho/xafcm.

https://doi.org/10.1016/j.patrec.2018.05.026
0167-8655/

means of a compressor. Let $C(x)$ be the number of bits used by a compressor to represent the string x. We will use a measure based on the notion of *relative compression* [4], denoted by $C(x||y)$, which represents the compression of x relatively to y. This measure obeys the following rules:

- $C(x||y) \approx 0$ iff string x can be built efficiently from y;
- $C(x||y) \approx |x|$ iff $K(x|y) \approx K(x)$.

Based on these rules, the **Normalized Relative Compression** (NRC) of the binary string x given the binary string y, is defined as

$$\mathrm{NRC}(x||y) = \frac{C(x||y)}{|x|}, \tag{1}$$

where $|x|$ denotes the length of x.

A more general formula for the NRC of string x, given string y, where the strings x and y are sequences from an alphabet $\mathcal{A} = \{s_1, s_2, \ldots s_{|\mathcal{A}|}\}$, is given by

$$\mathrm{NRC}(x||y) = \frac{C(x||y)}{|x| \log_2 |\mathcal{A}|}. \tag{2}$$

2. Extended-alphabet finite-context models

2.1. Compressing using extended-alphabet finite-context models

Let $\mathcal{A} = \{s_1, s_2, \ldots s_{|\mathcal{A}|}\}$ be the alphabet that describes the objects of interest. An extended-alphabet finite-context model (xaFCM) complies to the Markov property, i.e., it estimates the probability of the next sequence of $d > 0$ symbols of the information source (depth-d) using the $k > 0$ immediate past symbols (order-k context). Therefore, assuming that the k past outcomes are given by $x_{n-k+1}^{n} = x_{n-k+1} \ldots x_n$, the probability estimates, $P(x_{n+1}^{n+d} | x_{n-k+1}^{n})$ are calculated using sequence counts that are accumulated, while the information source is processed,

$$P(w|x_{n-k+1}^{n}) = \frac{\nu(w|x_{n-k+1}^{n}) + \alpha}{\nu(x_{n-k+1}^{n}) + \alpha |\mathcal{A}|^d} \tag{3}$$

where $\mathcal{A}^d = \{w_1, w_2, \ldots w_{|\mathcal{A}|}, \ldots w_{|\mathcal{A}|^d}\}$ is an extension of alphabet $\mathcal{A}$ to d dimensions, $\nu(w|x_{n-k+1}^{n})$ represents the number of times that, in the past, sequence $w \in \mathcal{A}^d$ was found having x_{n-k+1}^{n} as the conditioning context and where

$$\nu(x_{n-k+1}^{n}) = \sum_{a \in \mathcal{A}^d} \nu(a|x_{n-k+1}^{n}) \tag{4}$$

denotes the total number of events that has occurred within context x_{n-k+1}^{n}.

In order to avoid problems with "shifting" of the data, the sequence counts are performed symbol by symbol, when learning a model from a string.

Parameter α allows controlling the transition from an estimator initially assuming a uniform distribution to a one progressively closer to the relative frequency estimator.

The theoretical information content average provided by the ith sequence of d symbols from the original sequence x, is given by

$$-\log_2 P(X_i = t_i | x_{id-k}^{id-1}) \text{ bits}, \tag{5}$$

where $t_i = x_{id}, x_{id+1} \ldots x_{(i+1)d-1}$.

As testbed applications, we perform ECG biometric identification and compute a similarity measure between DNA sequences;After processing the first n symbols of x, the total number of bits generated by an order-k with depth-d xaFCM is equal to

$$-\sum_{i=1}^{n/d} \log_2 P(t_i | x_{di-k}^{di-1}), \tag{6}$$

Table 1
FCM representation of the sequence *AAABCC*.

Context c	$\nu(A\|c)$	$\nu(B\|c)$	$\nu(C\|c)$	$\nu(c) = \sum_{a \in \mathcal{A}} \nu(a\|c)$
BC	0	0	1	1
CA	1	0	0	1
AB	0	0	1	1
CC	1	0	0	1
AA	1	1	0	2

Table 2
Proposed xaFCM representation of the sequence *AAABCC* (with $d = 1$). Notice that this model has exactly the same information as the one in Table 1.

Context c			
BC	C: 1	Total: 1	
CA	A: 1	Total: 1	
AB	C: 1	Total: 1	
CC	A: 1	Total: 1	
AA	A: 1	B: 1	Total: 2

where, for simplicity, we assume that $n(modd) = 0$.

If we consider a xaFCM with depth $d = 1$, then it becomes a regular FCM with the same order k. In that sense, we can consider that a FCM is a particular case of a xaFCM.

An intuitive way of understanding how a xaFCM works is to think of it as a FCM which, for each context of length k, instead of counting the number of occurrences of symbols of $\mathcal{A}$, counts the occurrences of sequences $w \in \mathcal{A}^d$. In other words, for each sequence of length k found, it counts the number of times each sequence of d symbols appeared right after it.

Even though, when implemented, this might use more memory to represent the model, an advantage is that it is possible to compress a new sequence of length m, relatively to some previously constructed model, making only m/d accesses to the model. This significantly reduces the time of computation, as we will show in the experimental results presented in Sections 3 and 4.

Since, for compressing the first k symbols of a sequence, we do not have enough symbols to represent a context of length k, we always assume that the sequence is "circular". For long sequences, specially using small contexts/depths, this should not make much difference in terms of compression, but as the contexts/depths increase, this might not be always the case.

Since the purpose for which we use these models is to provide an approximation for the number of bits that would be produced by a compressor based on them, whenever we use the word "compression", in fact we are not performing the compression itself. For that, we would need to use an encoder, which would take more time to compute. It would also be needed to add some side information for the compressor to deal with the circular sequences – but that goes out of scope for our goal.

2.1.1. Example

Let x be the circular sequence *AAABCC*. Using a regular FCM with $k = 2$ and $\alpha = 0.01$, we would build the model from Table 1 to represent x.

It is easy to notice that this representation can be implemented using an hash-table of strings to arrays of integers with fixed size (alphabet size +1). However, we propose a different alternative, which consists of building a hash-table of hash-tables. The reason for doing so is that often the number of counts of symbols for each context is very sparse, which would be a waste of memory. To represent exactly the same model, we would build the structure presented in Table 2.