

DNA CODING USING FINITE-CONTEXT MODELS AND ARITHMETIC CODING

Armando J. Pinho, António J. R. Neves, Carlos A. C. Bastos and Paulo J. S. G. Ferreira

Signal Processing Lab, DETI / IEETA
University of Aveiro, 3810–193 Aveiro, Portugal
ap@ua.pt / an@ua.pt / cbastos@ua.pt / pjf@ua.pt

ABSTRACT

The interest in DNA coding has been growing with the availability of extensive genomic databases. Although only two bits are sufficient to encode the four DNA bases, efficient lossless compression methods are still needed due to the size of DNA sequences and because standard compression algorithms do not perform well on DNA sequences. As a result, several specific coding methods have been proposed. Most of these methods are based on searching procedures for finding exact or approximate repeats. Low order finite-context models have only been used as secondary, fall back mechanisms. In this paper, we show that finite-context models can also be used as main DNA encoding methods. We propose a coding method based on two finite-context models that compete for the encoding of data, on a block by block basis. The experimental results confirm the effectiveness of the proposed method.

Index Terms— DNA coding, source coding, finite-context modeling, bioinformatics, arithmetic coding.

1. INTRODUCTION

Recently, and with the completion of the human genome sequencing, the development of efficient lossless compression methods for DNA sequences gained considerable interest [1–7]. For example, the human genome is determined by approximately 3 000 million base pairs [8], whereas the genome of the wheat has about 16 000 million [9]. Since DNA is based on an alphabet of four different symbols (usually known as nucleotides or bases), namely, Adenine (A), Cytosine (C), Guanine (G), and Thymine (T), it takes approximately 750 MBytes to store the human genome (using $\log_2 4 = 2$ bits per symbol) and 4 GBytes to store the genome of the wheat.

In a previous work [10, 11], we proposed a three-state finite-context model for DNA protein-coding regions, i.e., for the parts of the DNA that carry information regarding how proteins are synthesized. Basically, this three-state model proved to be better than a single-state model, given additional

evidence of a phenomenon that is common in these protein-coding regions, i.e., a periodicity of period three.

More recently [12], we investigated the performance of finite-context models for unrestricted DNA, i.e., DNA including coding and non-coding parts. In that work, we have shown that a characteristic usually found in DNA sequences, the occurrence of inverted repeats, which is used by most of the DNA coding methods (see, for example, [4–6]), could also be successfully integrated in finite-context models. Inverted repeats are copies of DNA sub-sequences that appear reversed and complemented ($A \leftrightarrow T, C \leftrightarrow G$) in some parts of the DNA.

In this paper, we propose a lossless coding method for DNA sequences based on finite-context models and arithmetic coding. It uses two competing finite-context models that capture the statistical information along the sequence and, on a block basis, strive for encoding the data. For each block, the best of the two models is chosen, i.e., the one that requires less bits for representing the block. Moreover, we give experimental evidence that a correct tuning of the parameter controlling the Lidstone estimator (which is a generalization of the Laplace law of succession [13] and also contains the Jeffreys [14] / Krichevsky-Trofimov estimator [15] as a special case) is most relevant in the case of the higher order finite-context model. The experimental results obtained show that the proposed codec is able to give very competitive compression results and that, therefore, finite-context models can be used as the main method for lossless coding of DNA sequences.

This paper is organized as follows. In Section 2 we describe our algorithm, and in particular how we collect the statistical information needed by the arithmetic coding. In Section 3 we provide experimental results obtained with our method and we compare the results with one of the most recent specialized methods. Finally, in Section 4 we draw some conclusions.

2. THE PROPOSED METHOD

In this work, we propose a DNA lossless compression method that is based on two finite-context models of different orders that compete for encoding the data. Because DNA data are

This work was supported in part by the FCT (Fundação para a Ciência e Tecnologia) grant PTDC/EIA/72569/2006.

non-stationary, using two models with different orders allow a better handling both of DNA regions that are best represented by low order models and regions where higher order models are advantageous. Therefore, although both models are continuously updated, only the best one is used for encoding a given region. For convenience, the DNA sequence is partitioned into non-overlapping blocks of fixed size (one hundred DNA bases), which are then encoded by one (the best one) of the two finite-context models.

To help explain our algorithm, let us consider an information source that generates symbols, s , from a finite alphabet \mathcal{A} . At time t , the sequence of outcomes generated by the source is $x^t = x_1x_2 \dots x_t$. The proposed algorithm relies on a combination of two finite-context models that generate probability estimates that are then used for driving an arithmetic encoder [16–18] (see Fig. 1). Each model collects statistical information from a context of depth M_i , $i = 1, 2, M_1 \neq M_2$. At time t , we represent the two conditioning outcomes by $c_1^t = x_{t-M_1+1}, \dots, x_{t-1}, x_t$ and by $c_2^t = x_{t-M_2+1}, \dots, x_{t-1}, x_t$. Note that the total number of conditioning states of a model with context depth M (i.e., an order- M finite-context model) is $|\mathcal{A}|^M$. In the case of DNA, since $|\mathcal{A}| = 4$, an order- M model implies 4^M conditioning states.

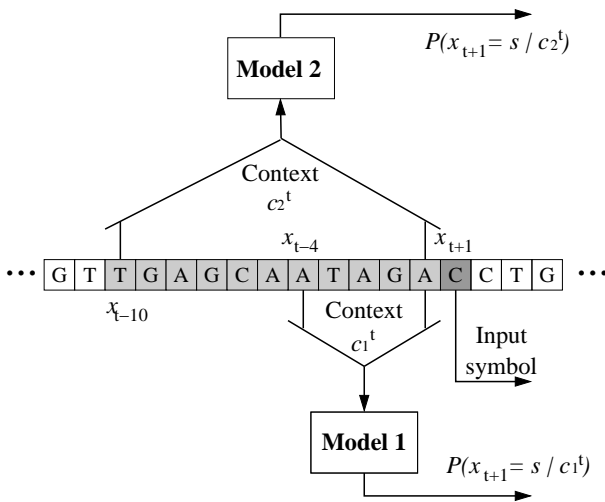


Fig. 1. Proposed model for estimating the probabilities: the probability of the next outcome, x_{t+1} , is conditioned by the M_1 or M_2 last outcomes, depending on the finite-context model chosen for encoding that particular DNA block. In this example, $M_1 = 5$ and $M_2 = 11$.

In practice, the probability that the next outcome, x_{t+1} , is s , where $s \in \mathcal{A} = \{A, C, G, T\}$, is obtained using the Lidstone estimator [19]

$$P(x_{t+1} = s | c^t) = \frac{n_s^t + \delta}{\sum_{a \in \mathcal{A}} n_a^t + 4\delta}, \quad (1)$$

Table 1. Simple example illustrating how finite-context models are implemented. The rows of the table represent a probability model at a given instant t . In this example, the particular model that is chosen for encoding a symbol depends on the last five encoded symbols (order-5 context).

Context, c^t	n_A^t	n_C^t	n_G^t	n_T^t	$\sum_{a \in \mathcal{A}} n_a^t$
AAAAA	23	41	3	12	79
⋮	⋮	⋮	⋮	⋮	⋮
ATAGA	16	6	21	15	58
⋮	⋮	⋮	⋮	⋮	⋮
GTCTA	19	30	10	4	63
⋮	⋮	⋮	⋮	⋮	⋮
TTTTT	8	2	18	11	39

where n_s^t represents the number of times that, in the past, the information source generated symbol s having c^t as the conditioning context. Parameter δ controls how much probability is assigned to unseen (but possible) events, and plays a key role in the case of high order models.¹ Note that Lidstone’s estimator reduces to Laplace’s estimator for $\delta = 1$ [13] and to the frequently used Jeffreys [14] / Krichevsky-Trofimov estimator [15] when $\delta = 1/2$. During our study, we found out experimentally that, using the proposed combination of two finite-context models, the probability estimates calculated for the higher order model lead to better compression results when smaller values of δ are used.

Usually, finite-context models are implemented by means of tables that collect the number of occurrences of a given alphabet symbol after some past sequence (the context). These counters are updated each time a symbol is encoded. Since the context template is causal, the decoder is able to reproduce the same probability estimates without needing additional information. However, because our method is composed of two models, one bit needs to be added to each block, indicating which one of the two finite-context models was used.

Table 1 shows an example, where an order-5 finite-context model is presented. Each row represents a probability model that is used to encode a given symbol according to the last encoded symbols (five in this example). Therefore, if the last symbols were “ATAGA”, i.e., $c^t = ATAGA$, then the model communicates the following probability estimates to the arithmetic encoder:

$$P(A|ATAGA) = (16 + \delta)/(58 + 4\delta),$$

$$P(C|ATAGA) = (6 + \delta)/(58 + 4\delta),$$

¹When M is large, the number of conditioning states, 4^M , is high, which implies that statistics have to be estimated using only a few observations.

Table 2. Table 1 updated after encoding symbol “C”, according to context “ATAGA”.

Context, c^t	n_A^t	n_C^t	n_G^t	n_T^t	$\sum_{a \in A} n_a^t$
AAAAA	23	41	3	12	79
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
ATAGA	16	7	21	15	59
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
GTCTA	19	30	10	4	63
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
TTTTT	8	2	18	11	39

$$P(G|ATAGA) = (21 + \delta)/(58 + 4\delta)$$

and

$$P(T|ATAGA) = (15 + \delta)/(58 + 4\delta).$$

The probabilities are then passed to the arithmetic encoder, which generates output bit-streams with average bit-rates almost identical to the entropy of the model [16–18]. The theoretical bitrate average (entropy) of a finite-context model after encoding N symbols is given by

$$H_N = -\frac{1}{N} \sum_{t=0}^{N-1} \log_2 P(x_{t+1} = s|c^t) \text{ bps}, \quad (2)$$

where “bps” stands for “bits per symbol”. Recall that the entropy of any sequence of four symbols is limited to two bps, a value that is achieved when the symbols are independent and equally likely.

According to the example of Table 1, and supposing that the next symbol to encode is “C”, we would require, theoretically, $-\log_2(6 + \delta)/(58 + 4\delta)$ bits to encode it. For $\delta = 1$, this is approximately 3.15 bits. Note that this is more than two bits because, in this example, “C” is the least probable symbol and, therefore, needs more bits to be encoded than the more probable ones. After encoding this symbol, the counters will be updated according to Table 2.

3. EXPERIMENTAL RESULTS

For the evaluation of the compression method described in the previous section we used the same DNA sequences used by Manzini *et al.* in [5], which are available from www.mfn.unipmn.it/~manzini/dnacorpus. This corpus contains sequences from four organisms: yeast (*Saccharomyces cerevisiae*, chromosomes 1, 4, 14 and the mitochondrial DNA), mouse (*Mus musculus*, chromosomes 7, 11, 19, x and y), arabidopsis (*Arabidopsis thaliana*, chromosomes 1, 3 and 4) and human (*Homo sapiens*, chromosomes 2, 13, 22, x and y).

Table 3. Compression values, in bits per symbol (bps), regarding a number of DNA sequences. The “DNA3” column shows the results obtained by Manzini *et al.* [5]. Column “FCM” contains the results of the proposed method. The orders of the two models that provided the best result are indicated under the columns labeled “ M_1 ” and “ M_2 ”.

Name	Size	DNA3 bps	FCM		
			M_1	M_2	bps
y-1	230 203	1.871	3	12	1.860
y-4	1 531 929	1.881	4	14	1.879
y-14	784 328	1.926	3	13	1.923
y-mit	85 779	1.523	5	9	1.484
Average	–	1.882	–	–	1.877
m-7	5 114 647	1.835	6	14	1.811
m-11	49 909 125	1.790	4	16	1.758
m-19	703 729	1.888	4	13	1.870
m-x	17 430 763	1.703	6	15	1.656
m-y	711 108	1.707	3	13	1.670
Average	–	1.772	–	–	1.738
at-1	29 830 437	1.844	6	16	1.831
at-3	23 465 336	1.843	6	16	1.826
at-4	17 550 033	1.851	6	15	1.838
Average	–	1.845	–	–	1.831
h-2	236 268 154	1.790	4	16	1.755
h-13	95 206 001	1.818	5	15	1.723
h-22	33 821 688	1.767	3	15	1.696
h-x	144 793 946	1.732	5	16	1.686
h-y	22 668 225	1.411	4	16	1.397
Average	–	1.762	–	–	1.711

Each of the sequences was encoded using the proposed model, choosing the best pair M_1, M_2 , such that $3 \leq M_1 \leq 8$ and $9 \leq M_2 \leq 18$. It was used the inverted repeats updating mechanism proposed in [12] and $\delta = 1$ for the lower order model and $\delta = 1/30$ for the higher order model. All information needed for correct decoding is included in the bit-stream and, therefore, the compression results presented in Table 3 account for that information. Besides the bitrate, the order of the models that provided the best results is also indicated in Table 3. For comparison, we include the results of the DNA3 compressor of Manzini *et al.* [5].

As can be seen from the results presented in Table 3, the proposed method, using two competing finite-context models, always provides better results than the DNA3 compressor. This confirms that the finite-context models can be successfully used as the main coding method for DNA sequences. Although we do not include here a comprehensive study of the impact of the δ parameter in the performance of the method, nevertheless we leave an indication of how it can influence the results. For example, using $\delta = 1$ for both models would

lead to bitrates of 1.869, 1.865 and 1.872, respectively for the “at-1”, “at-3” and “at-4” sequences, i.e. approximately 2% worse than when using $\delta = 1/30$ for the higher order model.

4. CONCLUSION

Finite-context modeling has been used for DNA compression only as a secondary, fall back method. However, as far as we are aware of, no systematic study of their potential has been carried out. In this paper, we have shown that a codec built of two competing finite-context models is indeed able to attain significant performance.

The experimental results show that the proposed approach can outperform the DNA3 coding method [5] for all DNA sequences used in our experiments. Although not the best method available in terms of compression performance, DNA3 is a well-balanced approach, with reasonable computation time requirements. Other methods, such as GeNML [6], can attain better compression results but at the cost of much longer processing times.

5. REFERENCES

- [1] S. Grumbach and F. Tahi, “Compression of DNA sequences,” in *Proc. of the Data Compression Conf., DCC-93*, Snowbird, Utah, 1993, pp. 340–350.
- [2] E. Rivals, J.-P. Delahaye, M. Dauchet, and O. Delgrange, “A guaranteed compression scheme for repetitive DNA sequences,” in *Proc. of the Data Compression Conf., DCC-96*, Snowbird, Utah, 1996, p. 453.
- [3] X. Chen, S. Kwong, and M. Li, “A compression algorithm for DNA sequences,” *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, pp. 61–66, 2001.
- [4] T. Matsumoto, K. Sadakane, and H. Imai, “Biological sequence compression algorithms,” in *Genome Informatics 2000: Proc. of the 11th Workshop*, A. K. Dunker, A. Konagaya, S. Miyano, and T. Takagi, Eds., Tokyo, Japan, 2000, pp. 43–52.
- [5] G. Manzini and M. Rastero, “A simple and fast DNA compressor,” *Software—Practice and Experience*, vol. 34, pp. 1397–1411, 2004.
- [6] G. Korodi and I. Tabus, “An efficient normalized maximum likelihood algorithm for DNA sequence compression,” *ACM Trans. on Information Systems*, vol. 23, no. 1, pp. 3–34, Jan. 2005.
- [7] B. Behzadi and F. Le Fessant, “DNA compression challenge revisited,” in *Combinatorial Pattern Matching: Proc. of CPM-2005*, Jeju Island, Korea, June 2005, LNCS, Springer-Verlag.
- [8] L. Rowen, G. Mahairas, and L. Hood, “Sequencing the human genome,” *Science*, vol. 278, pp. 605–607, Oct. 1997.
- [9] C. Dennis and C. Surridge, “*A. thaliana* genome,” *Nature*, vol. 408, pp. 791, Dec. 2000.
- [10] P. J. S. G. Ferreira, A. J. R. Neves, V. Afreixo, and A. J. Pinho, “Exploring three-base periodicity for DNA compression and modeling,” in *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, ICASSP-2006*, Toulouse, France, May 2006, vol. 5, pp. 877–880.
- [11] A. J. Pinho, A. J. R. Neves, V. Afreixo, Carlos A. C. Bastos, and P. J. S. G. Ferreira, “A three-state model for DNA protein-coding regions,” *IEEE Trans. on Biomedical Engineering*, vol. 53, no. 11, pp. 2148–2155, Nov. 2006.
- [12] A. J. Pinho, A. J. R. Neves, and P. J. S. G. Ferreira, “Inverted-repeats-aware finite-context models for DNA coding,” in *Proc. of the 16th European Signal Processing Conf., EUSIPCO-2008*, Lausanne, Switzerland, Aug. 2008.
- [13] P. S. Laplace, *Essai philosophique sur les probabilités (A philosophical essay on probabilities)*, John Wiley & Sons, New York, 1814, Translated from the sixth French edition by F. W. Truscott and F. L. Emory, 1902.
- [14] H. Jeffreys, “An invariant form for the prior probability in estimation problems,” *Proc. of the Royal Society (London) A*, vol. 186, pp. 453–461, 1946.
- [15] R. E. Krichevsky and V. K. Trofimov, “The performance of universal encoding,” *IEEE Trans. on Information Theory*, vol. 27, no. 2, pp. 199–207, 1981.
- [16] T. C. Bell, J. G. Cleary, and I. H. Witten, *Text compression*, Prentice Hall, 1990.
- [17] D. Salomon, *Data compression - The complete reference*, Springer, 2nd edition, 2000.
- [18] K. Sayood, *Introduction to data compression*, Morgan Kaufmann, 2nd edition, 2000.
- [19] G. Lidstone, “Note on the general case of the Bayes-Laplace formula for inductive or a posteriori probabilities,” *Trans. of the Faculty of Actuaries*, vol. 8, pp. 182–192, 1920.