CABIOS

Vol.9 no.4 1993
Pages 407 – 411

Discovering simple DNA sequences by the algorithmic significance method

Aleksandar Milosavljević[1] and Jerzy Jurka

Abstract

A new method, 'algorithmic significance', is proposed as a tool for discovery of patterns in DNA sequences. The main idea is that patterns can be discovered by finding ways to encode the observed data concisely. In this sense, the method can be viewed as a formal version of the Occam's Razor principle. In this paper the method is applied to discover significantly simple DNA sequences. We define DNA sequences to be simple if they contain repeated occurrences of certain 'words' and thus can be encoded in a small number of bits. Such definition includes minisatellites and microsatellites. A standard dynamic programming algorithm for data compression is applied to compute the minimal encoding lengths of sequences in linear time. An electronic mail server for identification of simple sequences based on the proposed method has been installed at the Internet address pythia@anl.gov.

Introduction

The basic idea behind the 'algorithmic significance' method is that the processes of discovery and inductive inference can be viewed as a search for patterns that lead to concise descriptions of observations. This principle, often referred to as Occam's Razor, has been invoked to justify the parsimony methods for reconstruction of phylogenetic trees and the minimal edit distance methods for sequence alignments.

Algorithmic significance is based on the minimal length encoding approach to pattern recognition and inference. The most general form of minimal length encoding has originally been proposed by Kolmogorov (1968), Solomonoff (1964) and Chaitin (1966). Cover and Thomas (1991) give a recent review of this general formulation, including its relationship to the Occam's Razor principle, data compression and Shannon entropy.

The algorithmic significance method is conceptually similar to the statistical significance method in the Neyman–Pearson hypothesis testing framework (e.g. Kiefer, 1987). The main difference is that the alternative hypothesis from the Neyman–Pearson framework is now represented by a decoding algorithm instead by a probability distribution. The null hypothesis is then refuted by exhibiting significantly shorter encodings of the observed data than expected based on the null hypothesis. This idea is closely related to the recently introduced concept of competitive encoding (Cover, 1991; Cover and Thomas, 1991).

In this paper we present an improved version of the algorithmic significance method, originally proposed in Milosavljević (1991). We then apply the method to discover simple DNA sequences, improving sensitivity of our original approach (Milosavljević and Jurka, 1991) to the same problem.

Many DNA sequences within primate and other genomes are simple in the sense that they consist of repeated occurrences of a few words. A single word may occur tandemly repeated many times, but in some cases the repetitive arrangement may involve multiple words and may be masked by the presence of random mutations, by interruptions and by other irregularities in the repetitive pattern. Certain kinds of such repetitive sequences have been referred to as mini- and microsatellites (Weber and May, 1989; Armour and Jeffreys, 1992), SSR (simple sequence repeat) sequences (Silver, 1992), and VNTR (variable number of tandem repeats) sequences.

Many simple sequences are highly polymorphic and thus very informative for genetic mapping (Weber and May, 1989; Goodfellow, 1992). Their rapid expansion was shown to cause several severe genetic diseases (Richards and Sutherland, 1992). It has been postulated that because of their potential for rapid mutation, such sequences may facilitate rapid adaptation by serving as a 'tune-up' mechanism for gene expression or for various biological functions in general (Trifonov, 1990).

In this paper we propose a method for defining simple sequences and an efficient algorithm for their extraction. The approach can be used at least at two levels. At the basic level, the algorithm can be applied to quickly identify and disassemble mini- and microsatellites into monomeric units. At a more advanced level it can be used for quantitative analyses of these repeats and for the analysis of the pattern of their distribution in sequenced DNA. We are not aware of any previous attempts to rigorously define simple sequences and to design exact and efficient algorithms for their extraction based on a rigorous definition.

Roughly, a sequence is defined to be simple if it can be encoded concisely by replacing occurrences of some words by pointers to the previous occurrences of the same words within

Linus Pauling Institute of Science and Medicine, 440 Page Mill Rd, Palo Alto, CA 94306, USA

[1]To whom reprint requests should be sent. Present address: Biological and Medical Research Division, Bldg 202, Argonne National Laboratory, Argonne, IL 60439-4833, USA

the same sequence. To obtain a more formal definition, we now apply some basic methods of data compression (Storer, 1988).

In an original pointer textual substitution data compression scheme (Storer, 1988), a repeated occurrence of a word is replaced by a pointer to its previous occurrence within the same sequence. We assume that a pointer consists of two positive integers: the first integer indicates the beginning position of a previous occurrence of the word while the second integer indicates the length of the word. For example, sequence

AGTCAGTTTT

may be encoded as

AGTC(1,3) (7,3)

where (1,3) points to the occurrence of AGT from position 1 to position 3, and (7,3) points to the occurrence of TTT from position 7 to position 9 in the original sequence.

The decoding algorithm consists of the following two steps:

1. replace each pointer by a sequence of pointers to individual letters;
2. replace the new pointers by their targets in the left-to-right order.

Continuing our example, the first step would yield

AGTC(1,1) (2,1) (3,1) (7,1) (8,1) (9,1)

and the second step would yield the original sequence. From this decoding algorithm it should be obvious that the original sequence can be obtained despite overlaps of pointers and their targets, as is the case with the pointer (7,3) in our example.

It pays off to replace a word by a pointer only if the letter-by-letter encoding of the same word is longer. To measure the encoding lengths precisely, we now assume the following encoding scheme for letters and pointers: each element (either a letter or a pointer) is encoded by a log 5-bit (where log denotes logarithm base 2) preamble that either specifies a letter or announces a pointer. In addition, a pointer also contains the 2 log n-bit encoding of the two positive integers that do not exceed the length n of the sequence. Thus, the total encoding length of a pointer is $\log 5 + 2 \log n$. Based on this encoding scheme, it pays off to replace an occurrence of a word of length k by a pointer to its previous occurrence if $k * \log 5 > \log 5 + 2 * \log n$. Thus, the length of the replaceable words grows with the logarithm of sequence length. [This is what one may have guessed based on the recent result in probability theory that states that the expected length of the longest common word between two random sequences is proportional to the logarithm of the length of the sequences (Arratia and Waterman, 1985).]

An encoding of a sequence may be conveniently represented by inserting dashes between letters and pointers and by replacing pointers by their targets. The encoding of our example would then be represented by

A−G−T−C−AGT−TTT

The algorithmic significance method

Let P_0 denote the distribution of probabilities over sequences under the assumption that they are generated at random. This distribution, usually referred to as null hypothesis, assigns probability $p_0(s)$ to a sequence s of finite length over an alphabet of size g. For example, the null hypothesis may state that every letter is generated independently with equal probability over all letters in the alphabet, in which case $p_0(s) = g^{-|s|}$, where $|s|$ denotes the length of sequence s.

Let A denote a decoding algorithm that is designed so that the sequences that contain certain patterns have short encodings. For example, sequences that contain a sufficient number of long repeated words can be concisely encoded using the decoding algorithm described in the previous section. By $I_A(s)$ we denote the length, in bits, of an encoding of s. The following theorem states that a sequence s is unlikely to have an encoding much shorter than $-\log p_0(s)$.

Theorem 1

For any distribution of probabilities P_0 over sequences and for any decoding algorithm A,

$$P\{-\log p_0(s) - I_A(s) \geq d\} \leq 2^{-d}$$

where $p_0(s)$ is the probability assigned to sequence s by distribution P_0, and where $I_A(s)$ is the length of an encoding of sequence s using algorithm A.

This theorem enables us to use algorithm A as an alternative hypothesis to refute the null hypothesis P_0 at the significance level 2^{-d}. Applying the inequality above to our example, the probability that a sequence s will have an encoding $d = 7$ bits less than $-\log p_0(s) = |s| \log g$ is less than 2^{-7}, which is less than the standard significance threshold of 0.01.

The exponential relationship between encoding length and probability is very useful in case when large sequence libraries are searched. For example, if the sequence library searched contains sequences of total length L, then to refute the null hypothesis at the significance level of 0.01 for any sequence in the library, $d = 7 + \log L$ bits would suffice.

Proof of theorem 1 follows.

Since the code specified by A is uniquely decodable, by Kraft−McMillan inequality, $\sum_s 2^{-I_A(s)} \leq 1$. Thus, there is a normalizing constant $b \geq 1$ such that $\sum_s b * 2^{-I_A(s)} = 1$. Now we define the alternative hypothesis as the probability distribution P_A that assigns probability $p_A(s) = b * 2^{-I_A(s)}$ to a sequence s. Assuming that $p_0(s)$ is positive for each sequence s, we define a random variable X_A that has value $p_A(s)/p_0(s)$ for a sequence s. Since X_A is always non-negative, by Markov inequality

$$P\{X_A \geq c\} \leq \frac{E_0[X_A]}{c}$$

where $c > 0$ and

$$E_0[X_A] = \sum_s p_0(s) \frac{P_A(s)}{p_0(s)} = 1$$

is the expectation of X_A based on the null hypothesis P_0. We also replace $p_A(s)/p_0(s)$ for X_A to obtain

$$P\left\{\frac{p_A(s)}{p_0(s)} \geq c\right\} \leq \frac{1}{c}$$

After taking logarithms

$$P\{-\log p_0(s) + \log p_A(s) \geq d\} \leq 2^{-d}$$

where $d = \log c$. After replacing $b * 2^{-I_A(s)}$ for $p_A(s)$

$$P\{-\log p_0(s) - I_A(s) + \log b \geq d\} \leq 2^{-d}$$

and finally, since $b \geq 1$

$$P\{-\log p_0(s) - I_A(s) \geq d\} \leq 2^{-d}$$

and this completes the proof of theorem 1.

The minimal length encoding algorithm

We now present an efficient algorithm for minimal length encoding, one of a large family of data compression algorithms that are based on textual substitution (Storer, 1988). The input is a particular sequence s in a finite alphabet and the encoding length $p \geq 1$ of a pointer. It is assumed, without loss of generality, that the encoding length of a letter is 1 (it is only the ratio between the pointer size and the encoding length of a letter that matters). The output of the compression algorithm is a minimal length of the sequence.

To present the algorithm, we first introduce some notation. Let n be the length of the sequence s. Let s_k denote the $(n - k + 1)$-letter suffix of s that starts at the kth position. Thus, we may write s_1 instead of s. Let $I(s_k)$ be the minimal encoding length of the suffix s_k (the pointers in this encoding may also point to the left of the kth position). Finally, let $l(i)$, where $1 \leq i \leq n$, denote the length of the longest word that starts at the ith position and that also occurs starting at a position $j < i$. If the letter at position i does not occur at any previous position, then $l(i) = 0$. Using this notation, we may now state the main recurrence (for a proof, see Storer, 1988):

$$I(s_i) = min(1 + I(s_{i+1}), p + I(s_{i+l(i)}))$$

This recurrence suggests that the minimal encoding length of $s = s_1$ can be computed in a linear-time right-to-left pass through the sequence, provided the values of $l(i)$, $1 \leq i \leq n$ are known in advance. The values of $l(i)$ can themselves be computed in an initial left-to-right pass in linear time by using a directed acyclic word graph data structure (Blumer *et al.*, 1985). Thus, the minimal length encoding of the whole sequence can be computed in linear time.

Implementation and application

The algorithm decribed above was implemented in C++ and is accessible via electronic mail. The electronic mail server reads the incoming mail messages that contain DNA sequences, identifies and analyzes simple segments, and then sends the results back to the sender. For more details on current availability and proper use of the server, send a word 'help' in the 'Subject' line to pythia@anl.gov. If you do not receive a satisfactory response, contact us directly at pythia-admin@anl.gov.

We now describe an example of the use of the program to identify simple sequences in genomic DNA. Throughout the example, the encoding length achieved by the encoding scheme described above is compared to the straightforward encoding scheme that uses 2 bits per letter and that corresponds to the null hypothesis that the letters are generated independently by a uniform distribution.

The complete DNA sequence of the Human Tissue Plasminogen Activator Gene (Friezner-Degen *et al.*, 1986), GenBank (Bilofsky and Burks, 1988) accession number K03021, containing 36 594 bases was searched for simple segments. The sequence was considered one window at a time, with the windows of length 128 and an overlap between adjacent windows of 64. The encoding length threshold of $22 \geq 7 + \log 36\,594$ bits was chosen so that the probability of any window having short encoding would be guaranteed not to exceed the value of 0.01.

Four simple segments were found. The first was the $(RY) * N$ run from position 7170 to position 7225 in the window 7169–7296; this window was encoded in 215 bits, $d = 41$ bits less than the 256 bits required by the straightforward encoding. The second was the $(AC) * N$ run followed by a $(TC) * N$ run in the window 16 897–17 024, also encoded in $d = 41$ bits less. The third was a poly(A) segment in the window 17 089–17 216, encoded in $d = 22$ bits less. The fourth was the $(TGATAGA) * N$ run from position 23 888 to position 24 458; Figure 1 contains the representation of an encoding of the window 24 321–24 448 in $d = 56$ bits less than required by the straightforward encoding.

The first and the fourth (and no other) simple segments were already annotated in the Release 69 of GenBank (Bilofsky and Burks, 1988). It may be interesting to note that the number of dinucleotide repeats in the second segment has recently been discovered to be polymorphic in the human population (Sadler *et al.*, 1991).

Discussion

In this paper we have attempted to provide an efficiently computable definition of a simple DNA sequence. We have also proposed 'algorithmic significance' as a general method for estimation of probability of random occurrence of a pattern in

Human Tissue Plasminogen Activator Gene, positions 24321 . . 24448

T-G-A-T-A-G-G-T-G-A-T-A-G-A-T-A-G-A-T-A-G-A-T-TGATAGAT-G-A-T-A-G-A-AGATTGATAGA
TGATAGA-T-A-C-ATAGGTGATAG-T-A-G-A-T-G-T-A-A-G-A-TGATAGATGATAGATA-GATAG
ATGATAGA-C-AGATTGATAGATGATAGA-G-A-G-A

Fig. 1. A sample of encodings.

DNA sequences, and we have applied it to discover simple DNA sequences. In the following we will make a comparison with some related methods.

Trifonov (1990) defines the notion of linguistic complexity C of a sequence of length n as $C = \prod_{i=1}^{n-1} U_i$, where the vocabulary size U_i is defined as the ratio between the number of distinct words of length i that occur in the sequence and the maximum possible number of distinct words of length i that a sequence of length n may possibly contain. The straightforward algorithm that checks the occurrences of all the subwords up to length $n - 1$ would require $\sum_{i=1}^{n-1} (n - i + 1) * i = \Theta(n^3)$ steps. Due to the large demand on computer time, only short windows of length 20 or so have been used in practice. A long sequence would be split into short windows; the complexities of individual windows would then be computed and averaged over the whole sequence.

The linguistic complexity approach is closely related to the approach presented in this paper because it also formally defines sequence complexity based on subword composition. However, linguistic complexity is more applicable to the analysis of global subword composition of large sequences, and even whole genomes. In contrast, the approach presented in this paper is most applicable to the problem of discovering patterns of length of several hundred letters or less within large sequences.

We may point out several general improvements over the linguistic complexity approach. First, the arbitrary assumption about word length is removed. Second, algorithms that are efficient even for arbitrary word length and arbitrary window size are introduced. Third, arbitrary measures of complexity and similarity are replaced by the minimal encoding length, a measure of the information content of sequences that has been studied extensively. Fourth, a simple but general criterion has been proposed for the assessment of significance of the discovered patterns.

Another contribution of this paper is the algorithmic significance method. In contrast to the asymptotic methods that require sophisticated study of distributional properties of each statistic of interest (e.g. longest repeated word, number of long repeats, etc.—Karlin *et al.*, 1989), the algorithmic significance approach can be applied to any kind of pattern, provided the pattern can be described in terms of an encoding scheme. Unlike the asymptotic methods, which provide only asymptotic estimates of statistical significance, the algorithmic significance approach provides an exact bound on the probability of random occurrence of a pattern. The algorithmic significance approach is also complementary to the methods that focus on particular statistics in the sense that it is concerned with 'global' properties of a sequence, instead focusing on the presence of particular patterns; for example, one may be interested in finding 'simple' sequences, in which case one would apply algorithmic significance, or one may be interested in significantly long repeated words in the sequences, in which case one would need to know the distributional properties of the length of the longest repeated word.

The most obvious direction for improvement of the proposed approach would be to apply the predictive and adaptive coding schemes that have been developed in the area of data compression (Williams, 1991). Such schemes generally achieve better compression in practice (e.g. Blumer, 1990), and thus may improve sensitivity. It would also be interesting to apply the algorithmic significance approach to other pattern-recognition problems that are hard to approach using standard statistical significance methods, e.g. to establish significance of sequence similarity based on subword composition or based on sequence alignment.

Acknowledgements

Professor Thomas Cover gave us the idea for improving an earlier version (Milosavljević, 1991) of the basic inequality for algorithmic significance. Discussions with Professors Emile Zuckerkandl and Edward Trifonov have greatly contributed to this work. This research has been supported by the DOE grant DE-FG03-91ER61152.

References

Armour,J.A.L. and Jeffreys,A.J. (1992) Recent advances in minisatellite biology. *FEBS Lett.*, **307**, 113–115.

Arratia,R. and Waterman,M.S. (1985) An Erdos–Renyi law with shifts. *Adv. Math.*, **55**, 13–23.

Bilofsky,H.S. and Burks,C. (1988) The GenBank® genetic sequence data bank. *Nucleic Acids Res.*, **16**, 1861–1864.

Blumer,A. (1990) Applications of DAWGs to data compression. In Capocelli,R.M. (ed.), *Sequences*. Springer-Verlag, Berlin.

Blumer,A., Blumer,J., Haussler,D., Ehrenfeucht,A., Chen,M.T. and Seiferas,J. (1985) The smallest automaton recognizing the subwords of a text. *Theor. Comput. Sci.*, **40**, 31–55.

Chaitin,G.J. (1966) On the length of programs for computing finite binary sequences. *J. Assoc. Comput. Machin.*, **13**, 547–569.

Cover,T.M. (1991) On the competitive optimality of Huffman codes. *IEEE Trans. Info. Theor.*, **37**, 172–174.

Cover,T. and Thomas,J. (1991) *Elements of Information Theory*. Wiley.

Freizner-Degen,S.J., Rajput,B. and Reich,E. (1986) The human tissue plasminogen activator gene. *J. Biol. Chem.*, **261**, 6972–6985.

Goodfellow,P.N. (1992) Variation is now the theme. *Nature*, **359**, 777–778.

Karlin,S., Ost,F. and Blaisdell,B.E. (1989) Patterns in DNA and amino acid sequences and their statistical significance. In *Mathematical Methods for DNA Sequences*. CRC Press, Boca Raton, FL.

Kiefer,J.C. (1987) *Introduction to Statistical Inference*. Springer-Verlag, Berlin.
Kolmogorov,A.N. (1968) Three approaches to the quantitative definition of information. *Int. J. Comput. Math.*, **2**, 157–168.
Milosavljević,A. (1991) Occam's razor cuts at the 8th bit: algorithmical significance, machine learning, and macromolecules. *Working Notes of the Workshop on AI Approaches to Classification and Pattern Recognition in Molecular Biology*.
Milosavljević,A. and Jurka,J. (1991) Discovering simple DNA sequences. *Working Notes of the Workshop on AI Approaches to Classification and Pattern Recognition in Molecular Biology*.
Richards,R.I. and Sutherland,G.R. (1992) Heritable unstable DNA sequences. *Nature Genet.*, **1**, 7–9.
Sadler,L.A., Blanton,S.H. and Daiger,S.P. (1991) Dinucleotide repeat polymorphism at the human tissue plasminogen activator gene (plat.). *Nucleic Acids Res.*, **19**, 6058–6058.
Silver,L.M. (1992) Bouncing off microsatellites. *Nature Genet.*, **2**, 8–9.
Solomonoff,R.J. (1964) A formal theory of inductive inference, part I. *Informat. Control*, **7**, 1–22.
Storer,J.A. (1988) *Data Compression: Methods and Theory*. Computer Science Press.
Trifonov,E.N. (1990) In Making sense of the human genome. *Human Genome Initiative and DNA Recombination*. Adenine Press, Vol. 1.
Weber,J.L. and May,P.E. (1989) Abundant class of human DNA polymorphisms which can be typed using polymerase chain reaction. *Am. J. Hum. Genet.*, **44**, 388–396.
Williams,R.N. (1991) *Adaptive Data Compression*. Kluwer.

Received on July 20, 1992; accepted on January 5, 1993