Calculating the Exact Probability of Language-like Patterns in Biomolecular Sequences

Kevin Atteson
atteson@peaplant.biology.yale.edu

Abstract

We present algorithms for the exact computation of the probability that a random string of a certain length matches a given regular expression. These algorithms can be used to determine statistical significance in a variety of pattern searches such as motif searches and gene-finding. This work improves upon work of Kleffe and Langebacker (Kleffe & Langbecker 1990) and of Sewell and Durbin (Sewell & Durbin 1995) in several ways. First, in many cases of interest, the algorithms presented here are faster. In addition, the type of pattern considered here strictly includes those of both previous works but also allows, for instance, arbitrary length gaps. Also, the type of probability model which can be used is more general than that of Sewell and Durbin, allowing for Markov chains. The problem solved in this work is in fact in the class of NP-hard problems which are believed to be intractable. However, the problem is fixed-parameter tractable, meaning that it is tractable for small patterns. The is problem is also computationally feasible for many patterns which occur in practice.

As a sample application, we consider calculating the statistical significance of most of the PROSITE patterns as in Sewell and Durbin. Whereas their method was only fast enough to exactly compute the probabilities for sequences of length 13 larger than the pattern length, we calculate these probabilities for sequences of up to length 2000. In addition, we calculate most of these probabilities using a first order Markov chain. Most of the PROSITE patterns have high significance at length 2000 under both the i.i.d. and Markov chain models. For further applications, we demonstrate the calculation of the probability of a PROSITE pattern occurring on either strand of a random DNA sequence of up to 500 kilo-bases and the probability of a simple gene model occurring in a random sequence of up to 1 megabase.

Introduction

Over the past few years, the analogy between DNA and human languages has been noted repeatedly(e.g. (Volker Brendel & Trifonov 1986; Dong & Searls 1994)). As with human languages, the "language" of DNA is often modeled stochastically(e.g. (Blaisdell 1985)). In contrast with human languages, however, DNA is often seen as consisting of long stretches which are random interspersed with sometimes language-like patterns such as protein sequence motifs, splice sites and open reading frames and genes. Hence, it is commonplace to search through long uncharacterized regions of DNA for these patterns. However, these elements may occur by chance in the random stretches of DNA, introducing false positives. In this paper, we present a general method for determining the probability of various patterns, namely patterns represented by regular expressions, a simple model of languages, occurring at random in long sequences. As an application of this work, we demonstrate the calculation of the probability of PROSITE (Bairoch 1995) patterns occurring in protein sequences of various length and the probability of regular expressions which represent simple models of genes.

Previous Work

Kleffe and Langbecker (Kleffe & Langbecker 1990) present a method for calculating the probability of patterns consisting of particular strings occurring within a long random sequence. This method requires an overhead of $O(km^2)$ and a running time of $O(kmn)$ where k is the alphabet size (e.g. 4 for nucleotides and 20 for proteins), m is the length of the pattern and n is the length of the random sequence. The methods presented in this paper, when applied to the problem addressed by Kleffe and Langbecker, have an overhead of $O(km)$ with a running time of $O(\min(kmn, m^3 \log n))$.

Sewell and Durbin (Sewell & Durbin 1995) present an algorithm for calculating the probability of a more general type of pattern, namely, the type of patterns used in the PROSITE database (Bairoch 1995), occurring beginning in the first n positions of a random sequence. This algorithm is worst case $O(\exp(\max(m, n)))$ where m is the length of the pattern. In fact, this running time limits the size of n which can be used and Sewell and Durbin only calculate the probabilities up to $n = 13$. The algorithm presented here works for more general types of patterns and runs in time $O(k \exp(m)n)$ and so is fixed-parameter tractable, that is, for small patterns, it is of linear order in the length of the sequence. Also, the running time is often much smaller than the worst case in practice.

An alternative approach to this problem is to use a sampling scheme to determine the probability within some bounds with a certain confidence (see, (Gore *et al.* 1997)). The approach presented here has the advantage of requiring only linear running time in the sequence length and so is amenable to long searches such as genome wide searches whereas the technique given in (Gore *et al.* 1997) is super-linear. Also, the deterministic approach has the advantage of yielding exact values and with certainty. However, the deterministic approach differs in that it requires a running time which is exponential in the length of the pattern in the worst case (in fact, it is #-P complete and so it is thought that there is no strictly polynomial time algorithm). Also, there are restricted forms of regular expressions which the present approach process in polynomial time and, in general, most expressions which occur in practice do not require the worst-case running time.

Introduction to Regular Languages

Regular Languages and Deterministic Finite Automata

The method presented here allows for the computation of the probability that a random string will match a given regular expression. Regular expressions are representations of a general type of pattern which includes, for example, patterns interspersed with arbitrary length gaps. We assume the reader is familiar with the basic theory of regular expressions and regular languages (Wood 1987). Let A^* denote the set of all finite strings over an alphabet A. A language is a subset of A^*. The regular expressions are a class of mathematical expressions used to denote members of the particular class of regular languages. We let ϵ denote the empty string. We otherwise use standard notation for regular expressions.

We also assume the reader is familiar with the basics of deterministic finite automata (DFA). In short, a DFA is a finite-state device for recognizing a regular language. Let Q denote the set of states of a DFA and $\delta : Q \times A \to Q$ its transition function, i.e. the function which determines which state it enters upon observing a particular input symbol from a particular state. We let $\delta^* : Q \times A^* \to Q$ denote the extended transition function, that is:

$$\delta^*(q,x) = \begin{cases} q & \text{if } x = \epsilon \\ \delta(\delta^*(q,y),a) & \text{if } x = ya \\ & \text{for some } z \in A \end{cases}$$

Examples of Regular Languages

The following examples illustrate some of the generality of patterns described by regular expressions. As a shorthand we use a period "." to denote the regular expression which represents any single character from the alphabet, e.g. for the alphabet $A = \{a,t,g,c\}$, we have "." is equivalent to $(a|t|g|c)$. Also, an expression followed by "$\{n,m\}$" where n and m are integers, denotes the expression repeated some number from n to m times.

1. $.*(R|K)(..|...)(D|E)(..|...)Y.*$ is the regular expression which represents any string containing prosite pattern PS00007.
2. $.*(c|a)aggt(a|g)agt.*(t|c)\{11,11\}.(c|t)agg.*$ is the regular expression which represents any string containing donor and acceptor consensus sites separated by an arbitrary length gap (e.g. a rough model for an intron).
3. $(a..|c..|g..|tc.|tt.|tac|tat|tgc|tgg|tgt)*$ is the regular expression which represents a string which is an open reading frame.

Computing the Probability of a Regular Language

Now let $X^n = X_1, X_2, \ldots, X_n$ be an i.i.d. (we later relax this assumption) random sequence from the alphabet A with probabilities $\text{Prob}(X_i = a) = p_a$ for $a \in A$. The key observation is that the sequence of states $\delta^*(s, X^i)$ forms a homogeneous Markov chain since each is determined from the previous state and the current symbol:

$$\begin{aligned} &\text{Prob}\left(\delta^*(s,X^{i+1}) = q_{i+1} \,\middle|\, \delta^*(s,X^i) = q_i,\right. \\ &\qquad \left.\delta^*(s,X^{i-1}) = q_{i-1}, \ldots, \delta^*(x,X^1) = q_1\right) \\ &= \text{Prob}\left(\delta(q_i, X_{i+1}) = q_{i+1}\right) \\ &= \sum_{\{a:\delta(q_i,a)=q_{i+1}\}} p_a \end{aligned}$$

Now let P be the matrix of transition probabilities, that is $P_{q,q'} = \text{Prob}\left(\delta^*(q, X_1) = q'\right)$. Let f be the vector such that $f_q = 1$ if $q \in F$ and $f_q = 0$ otherwise. The probability that the DFA recognizes X^n is given by $e_s^T P^n f$ where e_s denotes the *sth* row of the identity matrix. Note that the Markov chain is sparse, that is, for each q, the number of q' such that $P_{q,q'} > 0$ is at most $|A|$ since this set is equal to $\{\delta(q,a) : a \in A\}$. Hence, we can calculate the probability that the DFA recognizes X^n, i.e. $e_1^T P^n f$, with sparse matrix multiplication in $O(kpn)$ time where k is the alphabet size and p is the number of states of the DFA. If instead we use the power method (Stewart 1994) to find the *nth* power of P, we can find the probability that the DFA recognizes X^n in $O(kp^\alpha \log n)$ time and $O(p^2)$ space where α is the power in the exponent of the time required for matrix multiplication (e.g. using the naive algorithm $\alpha = 3$ and using Strassens' algorithm $\alpha = log_2 7$ but the latter algorithm is less numerically stable than the former (Bini & Pan 1994)). Note that this method for computing the probability is essentially the same as in Kleffe and Langebecker but over the more general set of all DFA's. This method can easily be generalized to the case when the process on sequences is a *rth*-order Markov chain by noting that the pairs $(\delta(s,X^n), X_n^{-r})$

is a Markov chain where X_n^{-r} denotes the sequence $X_{n-r+1}X_{n-r+2}\ldots X_n$.

Using the above observation, we can calculate the probability that a random string of length n matches a given regular expression by computing the DFA of the regular expression and multiplying the resulting sparse Markov chain transition matrix by the initial vector n times. However, the size of the matrix will equal the number of states of the DFA which can be exponential in terms of the size of the regular expression. Hence, the above method is only feasible for small regular expressions. In fact, the general problem of determining the probability that a string of length n is #-P complete (Gore *et al.* 1997) and so it is believed that no polynomial time algorithm for performing this computation exists. However, for many reasonable sized regular expressions, the DFA is not excessively large. Also, for some interesting subclasses of regular expressions, efficient algorithms exist for DFA construction. Given a DFA, it is possible to construct an equivalent DFA containing a minimal number of states in $O(p \log p)$ time (Alfred V. Aho & Ullman 1974). In many of the examples discussed below, this minimization was necessary to make the probability calculation feasible.

The Probability of Protein Motifs

We considered 3 models of random protein sequences. The first was the model used by Sewell and Durbin (Sewell & Durbin 1995) which we refer to as the Sanger model. The other two models were i.i.d. and *1st*-order Markovian built from all human protein sequences in the SwissProt data bank (A. & R. 1996). We have calculated the probability of PROSITE (release 13.0) patterns occurring in random sequences of up to length 2000. There were a total of 1154 entries in PROSITE, of which 1149 had machine-readable patterns associated with them. Of these, we were able to calculate the probabilities with the Sanger model for 1138. The i.i.d. model built from SwissProt showed no substantial differences from the Sanger model and so we performed most of the i.i.d calculations using the Sanger model. Of the 1138 patterns for which the probability was calculated under the Sanger model, we were able to calculate the significance of 1132 under the *1rst* order Markov chain since this requires an expansion of the DFA.

We considered a pattern significant if the probability of its occurrence in a random string of length 2000 was under 1%. Only the 18 PROSITE patterns given in Table were not significant under either the Sanger model or the Markov chain model. Of these 18 patterns, 16 were not significant under both models. PROSITE pattern PS00190, was significant under the Sanger model but not under the Markov model. Most of the difference in the probabilities for this pattern between the i.i.d. and Markov model disappears when using the i.i.d. model built from SwissProt in place of the Sanger model. Also, PROSITE pattern PS00339 was significant by a small margin under the i.i.d. model and insignificant by a small margin under the Markov model but the difference was slight.

In many cases, it is clear from the pattern that the PROSITE pattern is insignificant, for example, the N-linked glycosylation sites, with PROSITE pattern "N-P-[ST]-P", are widely known to be ambiguous. In other cases, it was more subtle. For instance, PS00013 with pattern "DERK(6)-[LIVMFWSTAG](2)-[LIVMFYSTAGCQ]-[AGS]-C" is not significant whereas PS00010 with pattern "C-x-[DN]-x(4)-[FY]-x-C-x-C" is.

The Probability of Patterns in DNA Sequences

The DNA Models

We built Markov chains of several orders on DNA based upon 2.1 megabases of data collected to model the statistics of introns and inter-genic regions in the GENSCAN gene finding program (Burge 1997). The DNA models were built using the same estimator as the protein models described earlier. For higher order Markov chains, the large number of parameters causes problems when To avoid these problems, we started each count with a "pseudo-count" of $\frac{1}{2}$. For example, the probability of nucleotide x given nucleotide sequence y is estimated by $\frac{N_{y,x}+\frac{1}{2}}{\sum_x N_{y,x}+2}$ where $N_{y,x}$ denotes the number of occurrences of the sequence yx.

In order to determine the order of Markov chain which was the best fit given the available data, we used the minimum description length principle(Rissanen 1978). This principle maintains that the model should be chosen with the order which allows us to describe the data most concisely, incorporating the length used to describe the model parameters.

Figure 1 shows the description length for the uniform i.i.d. model versus various order Markov chains including both general and strand-symmetric (that is, when we constrain the transition probabilities so that forward and reverse DNA sequences have the same probability) models. In all cases, the Markov chains were better models than the uniform and the strand-symmetric Markov chains were better than the general Markov chains. The order 7 strand-symmetric model had the lowest overall description length. This order is very high. It is generally acknowledged that the order of DNA is infinite and the minimum description length principle allows us to choose an order which trades off the amount of data and the number of parameters that need be trained. Unfortunately, these orders were too high to be useful for the methods to be discussed and we restrict ourselves to lower orders for what follows.

The Probability of Protein Motifs in Random DNA

Since most of the PROSITE sequences are significant if found in random protein, the question arises as to how significant are these sequences found in random DNA.

name	description	significance	
		i.i.d.	Markov
PS00005	Protein kinase C phosphorylation site.	1.0000	1.0000
PS00006	Casein kinase II phosphorylation site.	1.0000	1.0000
PS00008	N-myristoylation site.	1.0000	1.0000
PS00001	N-glycosylation site.	0.9996	0.9998
PS00004	cAMP- and cGMP-dependent protein kinase phosphorylation site.	0.9586	0.9798
PS00007	Tyrosine kinase phosphorylation site.	0.9496	0.9360
PS00009	Amidation site.	0.8241	0.8883
PS00002	Glycosaminoglycan attachment site.	0.4651	0.5795
PS00013	Prokaryotic membrane lipoprotein lipid attachment site.	0.2483	0.3451
PS00016	Cell attachment sequence.	0.3419	0.3326
PS00017	ATP/GTP-binding site motif A (P-loop).	0.1328	0.1324
PS00029	Leucine zipper pattern.	0.1257	0.1324
PS00215	Mitochondrial energy transfer proteins signature.	0.0241	0.0248
PS00190	Cytochrome c family heme-binding site signature.	0.0066	0.0196
PS00343	Gram-positive cocci surface proteins 'anchoring' hexapeptide.	0.0163	0.0195
PS00225	Crystallins beta and gamma 'Greek key' motif signature.	0.0182	0.0154
PS00342	Microbodies C-terminal targeting signal.	0.0135	0.0119
PS00339	Aminoacyl-transfer RNA synthetases class-II signature 2.	0.0135	0.0093

Table 1: PROSITE patterns which are insignificant under the protein models.

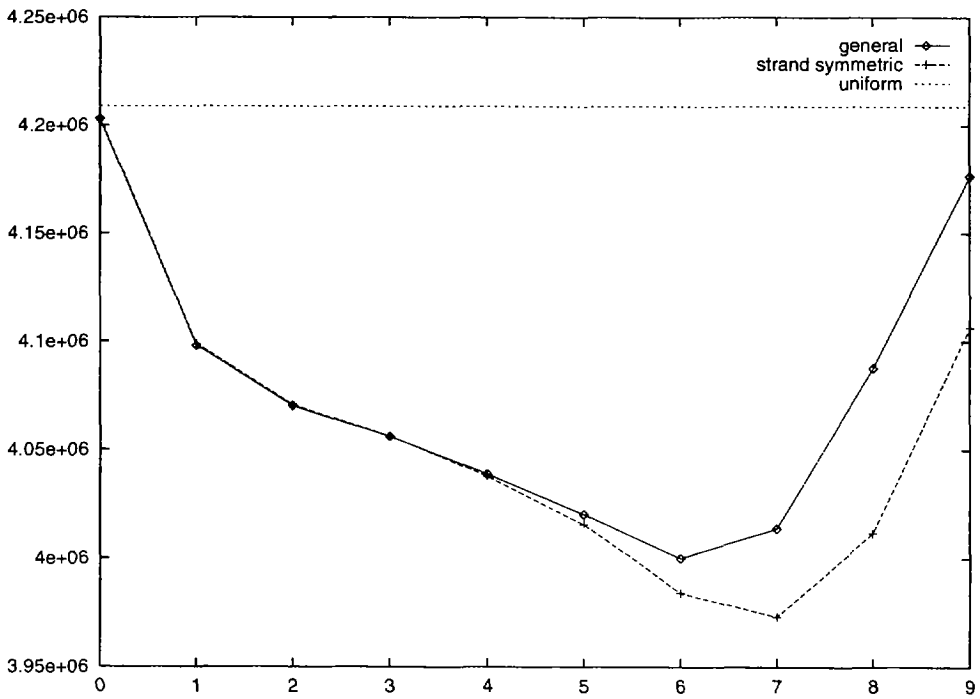

Figure 1: The minimum description length of the uniform, strand symmetric and general Markov chains of varying order.

The question is important because the significance of the motif might suggest the presence of a gene.

As an example of constructing a DNA regular expression, consider PROSITE pattern PS00007 discussed in item 1 of Section . To construct a regular expression which represents the presence of this pattern on one strand of the DNA, it is necessary to convert the amino acid residues to the corresponding codons:

$$.*(cg.|ag(a|g)|aa(a|g))(.....|.....)$$
$$(ga(t|c)|ga(a|g))(.....|.....)ta(t|c).*$$

In order to construct a regular expression representing the presence of this pattern on either strand, one must in addition include the alternative of the reverse complement of the above string in the expression. However, this alternative causes a substantial increase in the number of states of the resulting DFA.

Figure 2 shows the probability of the PROSITE pattern PS00010, occurring on either the forward or reverse strand of random DNA with varying orders of Markov dependence. The differences between the various Markov chain orders are not substantial. The pattern is significant at 1% out to about 34 kilo-bases and at 5% out to about 174 kilo-bases.

While calculating the probability of occurrence on both strands of the DNA causes the DFA to grow larger, this probability can be approximated from the probability of its occurrence on a single strand. If the occurrences of the pattern on one strand and the reverse strand are independent, then the probability of the occurrence on either strand is $1-(1-p)^2$ where p is the proability of occurrence on a single strand. While these events are not exactly true, they should hold to a high degree of approximation. Figure 3 shows the probability of PS00010 occurring on either strand of random DNA under an i.i.d. model versus the estimated probability based upon the probability of its occurrence on a single strand. In fact, the curves are indistinguishable.

Simple Gene Models

Using regular expressions, it is possible to evaluate the probability of various signals for genes occurring at random. Consider the rough model of an intron given by item 2 of Section . In practice, parts of these signals are weak and would be better modeled by allowing substitutions but we keep the expression simple for the purpose of demonstrating the technique. More sophisticated models are easily incorporated. Figure 4 shows the probability of various numbers of introns, modeled by item 2 of Section , occurring in random DNA sequences using a *2nd* order Markov chain. Note that even the occurrence of 3 introns is only significant at 5% for random sequence up to 107 kilo-bases.

While the presence of donor and acceptor pairs in a long random sequence is not significant, in real genes, one of the 3 reading frames between an acceptor and a successive donor will be an open reading frame. Let S be a shorthand for the open reading frame regular expression given by item 3 of Section . Again using the consensus sequences for splice sites, an internal exon can be represented by the following regular expression:

$$(t|c)\{11,11\}.(c|t)agg(\epsilon|.|..)S(\epsilon|.|..)(c|a)aggt(a|g)agt$$

Figure 5 shows how the probability of varying numbers of introns when the sequences between introns are constrained to have an open reading frame and the first exon is started with a start codon and an open reading frame. A single such intron remains insignificant but more than one becomes significant. This is because the acceptor and following donor sites must have an open reading frame spanning the intervening space which essentially limits the distance by which these can be separated.

Figure 6 shows the probability changes with different model orders. The probability of a match to the simple gene model dramatically increases with the order of the Markov chain. One possible explanation for this phenomenon is that since part of the data that we are using is from introns and so perhaps the model learns features of the introns such as splice sites. However, removing the flanking splice sites, including the pyrimidine-rich region involved in the acceptor site, does not mitigate this effect. It is possible that there are alternative splicing sites within the introns which bias the statistics.

Acknowledgments

Thanks to Chris Burge and Jonathan Schug for useful suggestions and data.

References

A., B., and R., A. 1996. The SWISS-PROT protein sequence data bank and it new supplement TrEMBL. *Nucleic Acids Research* 24:21–25.

Alfred V. Aho, J. E. H., and Ullman, J. D. 1974. *Design and analysis of computer algorithms.* Addison-Wesley Publishing Company.

Bairoch, A. 1995. The prosite database, its status in 1995. *Nucleic Acids Research* 24(1):189–196.

Bini, D., and Pan, V. 1994. *Polynomial and Matrix Computations.* Birkhäuser.

Blaisdell, B. E. 1985. Markov chain analysis finds a significant influence of neighboring bases on the occurrence of a base in eukaryotic dna sequences. *Journal of Molecular Evolution* 21:278 288.

Burge, C. 1997. *Identification of Genes in Human Genomic DNA.* Ph.D. Dissertation, Stanford University.

Dong, S., and Searls, D. 1994. Gene structure prediction by linguistic methods. *Genomics* 23(3):540–551.

Gore, V.; Jerrum, M.; Kannan, S.; Sweedyk, Z.; and Mahaney, S. 1997. A quasi-polynomial-time algorithm for sampling words from a context-free language. manuscript.

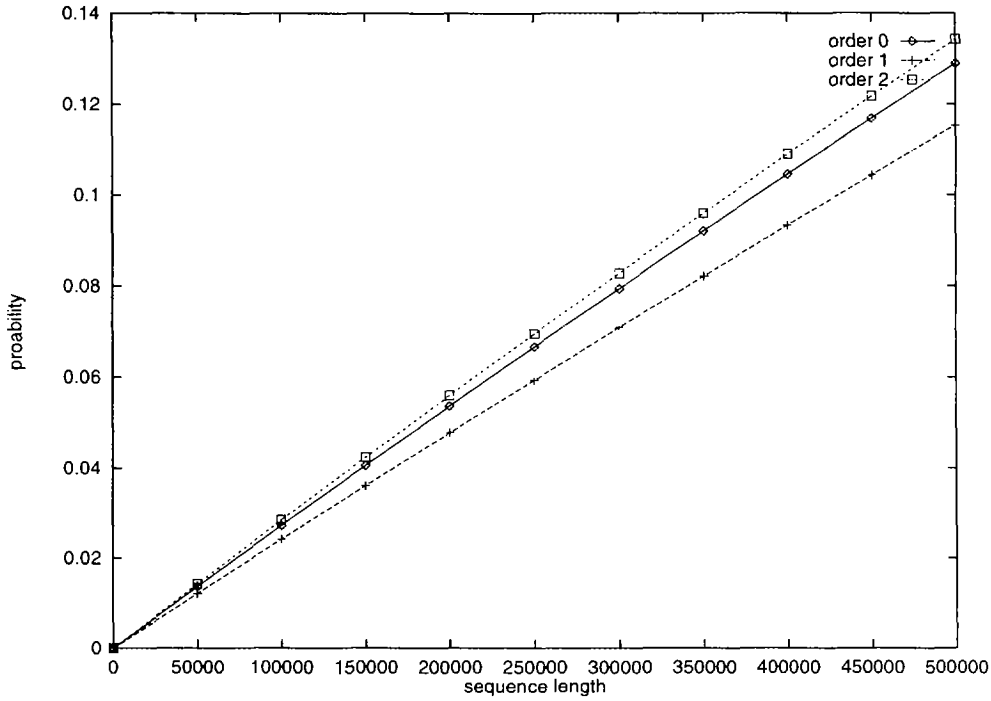

Figure 2: The probability of PROSITE pattern PS00010 occurring on either the forward or reverse strand of random DNA under Markov chains of several orders.

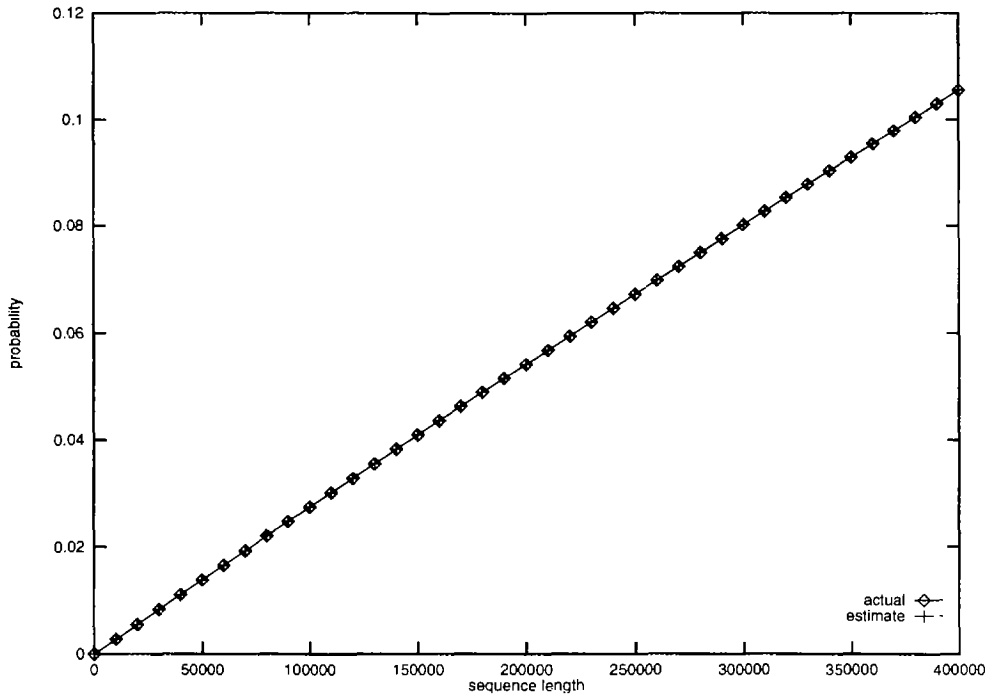

Figure 3: The probability of PROSITE pattern PS00010 occurring on either strand of random DNA under an i.i.d. model versus that estimated from its probabilities of occurrence on a single strand.

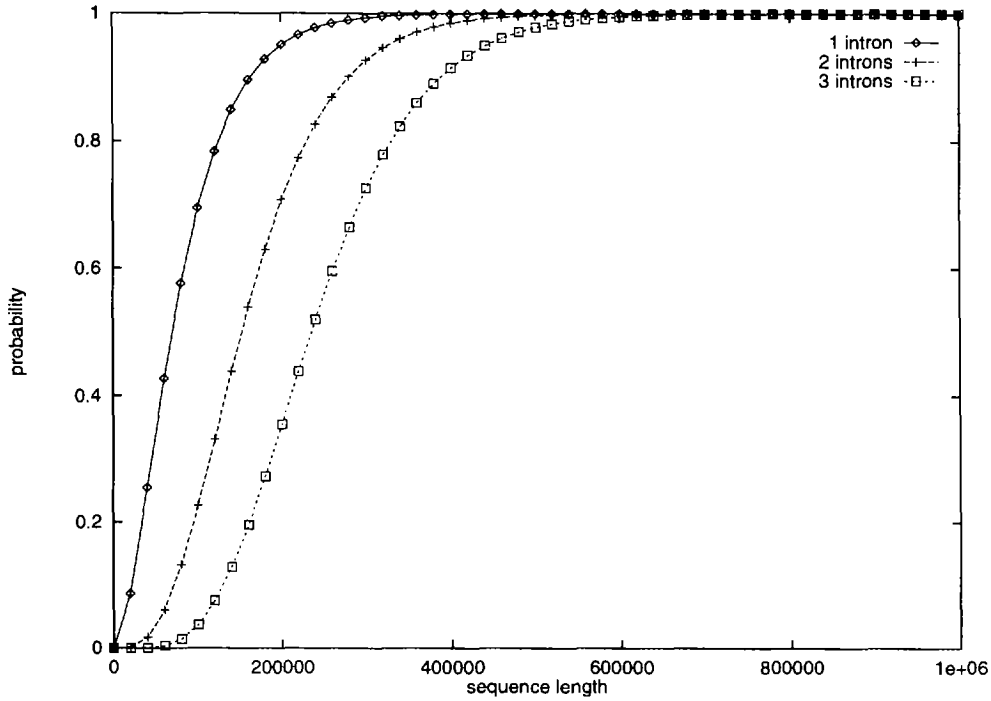

Figure 4: The probability of various numbers of introns occurring in random DNA sequence from a *2nd* order Markov chain.

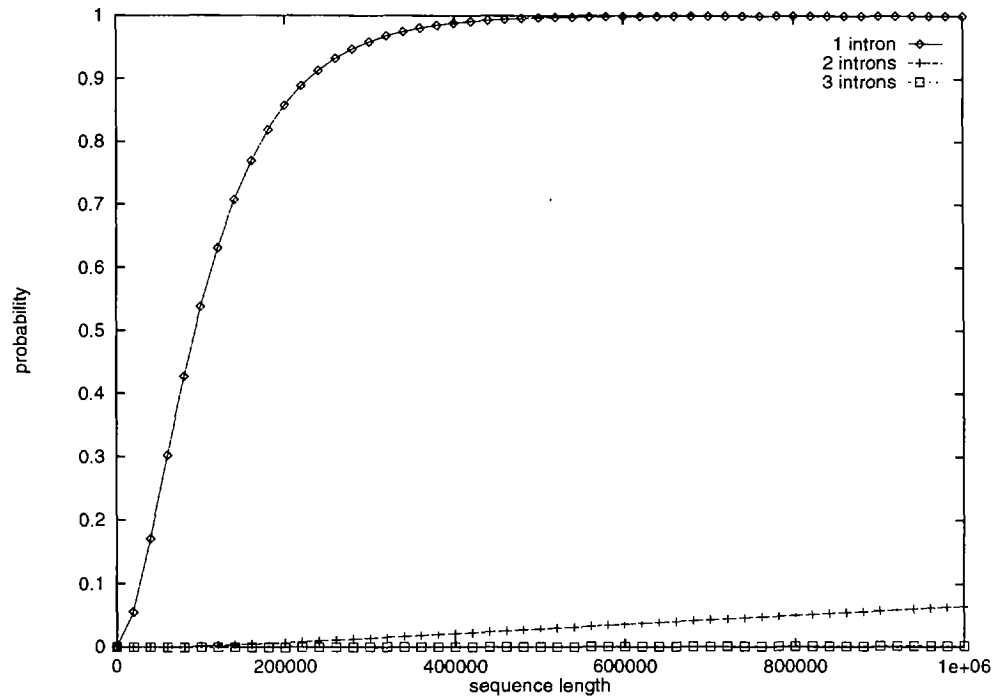

Figure 5: The probability of various numbers of introns each preceded by an exon in random DNA sequence from a *2nd* order Markov chain.

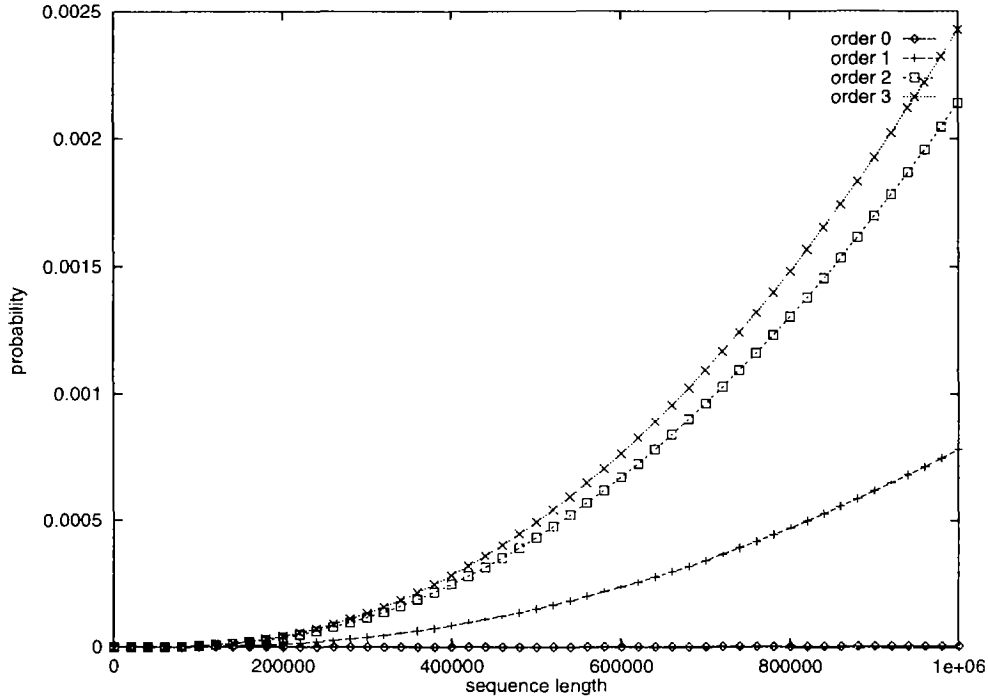

Figure 6: The probability of 3 introns with preceding exons in random DNA sequence from varying order Markov chains.

Kleffe, J., and Langbecker, U. 1990. Exact computation of pattern probabilities in random sequences generated by markov chains. *CABIOS* 6(4):347–353.

Rissanen, J. 1978. Modelling by shortest data description. *Automatica* 14:465–471.

Sewell, R. F., and Durbin, R. 1995. Method for calculation of probability of matching a bounded regular expression in a random data string. *Journal of Computational Biology* 2(1):25 31.

Stewart, W. J. 1994. *Introduction to the Numerical Solution of Markov Chains.* Princeton University Press.

Volker Brendel, J. S. B., and Trifonov, E. N. 1986. Linguistics of nucleotide sequences:morphology and comparison of vocabularies. *Journal of Biomolecular Structure and Dynamics* 4(1):11–21.

Wood, D. 1987. *Theory of Computation.* John Wiley & Sons, Inc.