SOME CHALLENGES IN COMPUTATIONAL BIOLOGY

M. Vidyasagar

Advanced Technology Centre
Tata Consultancy Services
6th Floor, Khan Lateefkhan Building
Hyderabad 500 001, INDIA
sagar@atc.tcs.co.in

Keywords: Computational biology, string alignment, hidden Markov models

Abstract

In this paper, we review some of the many challenging problems in computational biology that are amenable to treatment using a systems approach. Specific problems discussed include string alignment and protein structure prediction.

1 Introduction

Recent advances in experimental biology have resulted in both a substantial increase in the *speed* of experimentation as well as a significant reduction in the *cost* of experimentation. As a result, it is now possible to generate "raw data" at a far greater rate than was possible a few years ago, and at very little cost. However, in order to be useful, this data needs to be turned into "information." This is the aim of a discipline known earlier as "computational biology," and more recently as "bioinformatics."

The mathematical and computational problems associated with biology have attracted the attention of some top-notch mathematicians for several decades now. Perhaps one of the best examples is S. Karlin, whose book "Stochastic Processes" published in the 1960's already refers to the relationships between Markov chains and biological problems; see [7], p. 37. Karlin is also a co-inventor of a widely used probabilistic method known as BLAST (Basic Linear Alignment Search Technique) for aligning two strings defined over a common alphabet [8, 9, 4]. Thus, while there has been a long-standing tradition of studying biological problems using mathematical approaches, the recent excitement about advances in experimental biology has substantially enhanced the interest of the mathematical and computer community in the subject of computational biology. It is by now realized that, unless some significant advances are made in this area, much of the promise of biology will remain unrealized. This is exemplified by the fact that, whereas earlier biology was considered to be almost exclusively an experimental science, nowadays it is thought of as both an experimental as well as an information-based science.

The aim of this paper is to present a *personal* view of some challenging problems in computational biology, that are easily expressed in terminology and notation that is readily accessible to applied mathematicians, and in particular the control theory community. It is the belief of the author that the problems presented here are both of immediate interest to the biological community and also somewhat tractable. Due to limitations of space, only a few problems are presented here. However, there are many other problems meeting both of the above criteria that could be included.

2 A Brief Introduction to Some Aspects of Biology

A person with a mathematical training and a mathematical bent of mind will be immediately struck by the fact that *biologists think very differently from mathematicians*. A well-known Indian biologist told the author with a straight face that "Biology is today where physics was in the sixteenth century." Probably that is an exaggeration, but there is no doubt that many of the fundamental governing principles of biology are still being discovered. This is one of the reasons for the rapid obsolescence of biological textbooks and papers. While it is common in mathematics (or control theory) to revisit a thirty or forty year-old book or paper, most biologists would not bother reading anything that is more than a few years old. Though biologists, like all scientists, are committed to a reductionist approach to their science (whereby the subject can be explained by a few foundational principles governing everything else), they are somewhat handicapped by the large number of exceptions that need to be made to any putative theory, except in a few specific situations. Thus, for an applied mathematician, a natural starting point would be those aspects of biology that have in some sense "stabilized" and can be explained in terms of something resembling the axiomatic approach that is common in mathematics. The choice of topics discussed in this paper is governed by this criterion, namely, that they can be explained in an axiomatic manner. However, it is worth repeating again that there are several other problems besides those mentioned here that would meet this criterion.

This section contains a very high level description of some relevant notions of biology and the associated computational problems. Detailed descriptions of the problems, including the state of the art and future challenges, are given in later sections.

2.1 Genomics

All living things are made of DNA, or deoxyribonucleic acid. DNA is arranged in the famous double helix configuration dis-

covered by Watson and Crick exactly fifty years ago. Thus there are two polymers, or chains, consisting of the basic building blocks, namely nucleotides. Each chain is composed of four types of nucleotides, each characterized by four bases: Adenine (A), Cytosine (C), Guanine (G), and Thymine (T). Thus one can think of each strand in the double helix as a *huge* string over the four-symbol alphabet $\{A, C, G, T\}$. Each strand of the helix of the DNA has a definite beginning and an end; it cannot be "read backwards." The starting side is referred to as the $5'$ end and the ending side is referred to as the $3'$ end. The fact that there is a definite spatial ordering to the DNA strand allows us to model the *spatial* ordering as a *temporal* ordering and apply the methods of time series analysis. Because of the chemical bonding, if one side of the double helix contains an A at a particular location, the corresponding location on the other side *must* contain a T, and vice versa. Similarly, C and G are paired. The consequence is that once we know the string describing one strand of the double helix, we can unambiguously determine the other strand.

The description of one strand of the DNA as a string over $\{A, C, G, T\}$ is referred to as the **genome** of the organism. The length of the genome varies widely across life forms. The table below shows the lengths of the genomes of various organisms.

Table 1. Typical Lengths of the Genomes of Various Organisms

Type	Organism Name	Length of Genome
Virus	HIV Type 2	8,016
Bacterium	Escherichia Coli	$\approx$ 4.2 million
Mammal	Mouse	$\approx$ 2.4 billion
Mammal	Homo Sapiens	$\approx$ 3.2 billion
Plant	Rice	$\approx$ 10 billion

A few points are worth noting about this table.

1. The genome is each organism has a very precise and unvarying length. Thus each HIV Type 2 virus has *exactly* 8,016 nucleotides – it cannot have 8,017 or 8,015 for example. However, two viruses of the same type need not be identical at all locations. The variations at individual locations are referred to as **Single Nucleotide Polymorphisms (SNP's)**. Because the lower level organisms have had their genomes sequenced many times over, the exact length of their genome sequence is known. Thus the length of the *E coli* genome is known precisely, even though it shown as approximately 4.2 million in the above table. On the other hand, the human genome has not been sequenced sufficiently many times for us to know the precise length.

2. The above table shows that there is a rough correlation between what we perceive to be the "intelligence" or "complexity" of the organism and the length of its genome sequence. However, our smug feeling of superiority is shattered when we realize that the rice genome is about three times longer than ours. Thus it is clear that "length of the genome" does not *directly* translate into "complexity of behaviour."

3. The mouse genome has about 80% overlap with the human genome. The chimpanzee genome has about 98% overlap with the human genome. Thus, by studying the genomes of these organisms and (if necessary) by carrying out laboratory experiments, we can aspire to obtain a better understanding of the human genome.

4. The genomes of two humans differ in only about 0.1% of the locations, i.e., in about three million locations. These local variations are referred to as SNP's, as mentioned above. Finding out how SNP's affect a person is one of the major computational challenges.

2.2 Genes and Coding Regions

The process of life requires that DNA becomes RNA which in turns produces several proteins. For the present purposes, it is enough to think of RNA as another string over the alphabet of RNA nucleotide symbols $\{A, C, G, U\}$. Thus, so far as we are concerned, the symbol T has been replaced by U. Triplets of RNA nucleotide symbols are referred to as "codons." Clearly there are $4^3 = 64$ codons. Out of these 64 codons, three are called stop codons, while the remaining 61 codons produce 20 amino acids. A table giving the correspondence between the 61 codons and the 20 amino acid symbols, as well as sketches of the shapes of the 20 amino acids, can be found in [2], pp. 137-138. It is therefore apparent that there is a redundancy in the map between the 64 codons and the 21 amino acid plus STOP symbol. However, to date no plausible explanation has been advanced for the logic behind this map.

Among the 61 codons that produce 20 amino acids (which are the building blocks of proteins), some are called "start" codons. Of course, as stated above, the three codons that don't produce any amino acids are called stop codons. In "lower" life forms known as Prokaryotes, the RNA consists of a series of genes, which consist of "coding" and "non-coding" regions. The length of the coding region is a strict multiple of three; that is, each coding region consists of a number of codons. Each of these codons is mapped into the corresponding amino acid. The non-coding regions do not produce any amino acids. In "higher" forms of life known as Eukaryotes (which includes humans), there is an added complication. Each gene consists of a series of "introns" that intersperse the "exons" that actually correspond to the coding plus non-coding regions in a Prokaryote. When the RNA gets replicated, the introns seem to get cut out and only the exons replicate themselves; this phenomenon has no parallel in the case of Prokaryotes. Therefore, when two different Eukaryote organisms (e.g., a mouse and a man) have similar genes with similar functionalities, the exons have a very close resemblance to each other when viewed as strings over the alphabet $\{A, C, G, U\}$. However, the introns across organisms need not bear any resemblance to each other.

There is still some ambiguity as to where a coding region starts

(though not about where it ends). To put it simply, every coding region must start with a start codon, but every start codon does not signify the start of a coding region. The parts of the RNA that meet some simple necessary conditions (e.g., they start with a start codon and end with a stop codon, there is only one stop codon in the entire stretch, and that is at the end, the length of the string is not too short, etc.) is referred to as an **Open Reading Frame (ORF)**. One of the important problems in genomics is to determine whether an ORF is really a coding region or not. Note that some authors refer to an ORF as a "putative gene."

To summarize, we can state two very "high level" problems that can be tackled using mathematical techniques.

1. Given a gene, is it possible to distinguish the coding regions from the non-coding regions?

2. Given an ORF or a putative gene, is it possible to compare this string with another string that is *known* to be a gene, to determine the degree of similarity between the strings?

2.3 Proteins

On the next higher step up the complexity ladder are proteins. As stated above, at a very basic level a protein is just a sequence of amino acids. Thus, just as a genome can be thought of as a long string over the four-symbol alphabet $\{A, C, G, T\}$, a protein can be thought of as a string over a twenty symbol alphabet of the amino acid symbols. Unlike the genome sequence which can be billions of symbols long, or a gene sequence which can be tens of thousands of symbols long, a protein is a string that is on average about 250 amino acid symbols long. The shortest proteins consist of just about 50 amino acids, whereas very long proteins consist of up to 2,000 amino acids or even more. In biology, the description of a protein in terms of its amino acid sequence is called the **primary structure**. It is now accepted in biology that the primary structure of a protein determines *everything* about the protein, such as its three-dimensional shape, chemical activity, etc. However, actually translating the primary structure into this information poses a formidable challenge.

In principle, a protein folds up in three-dimensional space into a configuration that minimizes its total potential energy. After all, even a protein must obey the laws of physics! However, the expression for the potential energy is extremely complicated, especially if one takes into account the interactions between the various molecules that comprise the protein and its surroundings (usually water). Moreover, the energy function seems to have a huge number of non-global local minima in configuration space. Thus finding the three-dimensional structure of a protein by directly minimizing the energy function has not proved fruitful except for relatively short proteins. On the other side, it is possible to determine the 3-D structure of a protein using experimental techniques, either NMR for short proteins or particle accelerator experiements for long proteins. Clearly this is an expensive route to determining structure, and it would be preferable to find some other methods that work at least *most* of the time, even if not always. It can be said fairly that in *industry* (as opposed to academia), the ultimate purpose of bioinformatics is to discover new drugs, because that is where the money is. Drug discovery consists, in very simple terms, of determining which protein needs to have its activity suppressed or enhanced, and which chemical molecule will do the job. For this purpose, it is often enough to describe a protein in terms of its so-called secondary structure. In the secondary structure the local shape of a protein is described as one of just three entities, denoted by the symbols α, β, δ. These symbols are referred to respectively as helix, sheet, and coil. Note that there is no simple relationship between the length of a protein sequence (i.e., the length of its primary structure) and the length of its secondary structure.

There are in effect two distinct approaches to protein structure prediction. The first is called "homology" prediction (but the "homology" spoken of here has nothing to do with algebraic geometry!). In this approach, one starts with the primay structure of a protein whose 3-D structure he wishes to determine, and seeks a protein of known 3-D structure whose own primary sequence "closely" matches that of the protein at hand. A rule of thumb states that if two proteins have overlap of about 40%, then their 3-D structures are similar. Of the roughly 100,000 naturally occuring proteins, roughly 19,000 have had their 3-D structure determined, and naturally the number is constantly increasing. The purpose of homology-based modelling is to determine the structure of a protein of interest (which could either be "natural" or artificially constructed) by comparing it with a protein of known structure. The second approach is called "ab initio" prediction. The approach here is to map the primary structure directly into the secondary structure, using a variety of approaches. Neural networks have been used with *some* success to predict the secondary structure of a protein from its primary structure. See [2] for details. In spite of this, the problem of predicting protein structure can still be considered to be "open."

3 String Alignment

From the preceding discussion, it is clear that a central problem in computational biology is *string alignment*. This problem arises in at least two contexts: (i) Matching an RNA fragment with another that is known to be a gene, to determine the similarity if any. (ii) Matching an amino acid sequence with another whose 3-D structure is known to determine the similarity if any. While *biologically* the two problems are quite distinct, from a *computational* standpoint both problems are quite similar. A general problem formulation that encompasses both can be stated as follows:

Suppose $\mathcal{A}$ is the alphabet of interest (either the four-symbol nucleotide alphabet or the twenty-symbol amino acid alphabet), and suppose $\mathbf{u}, \mathbf{v}$ are two strings over $\mathcal{A}$. Suppose to be specific that $\mathbf{u}$ is shorter than $\mathbf{v}$. It is straight-forward to determine whether $\mathbf{u}$ is a *perfect* substring of $\mathbf{v}$. For this purpose,

one can set up a finite state machine with input that stops if and only if it encounters the input stream $\mathbf{u}$. Text editors do this kind of thing routinely. A slight variation of this approach can be used to determine the longest substring of $\mathbf{u}$ that is a perfect substring of $\mathbf{v}$.

However, in biological applications it will rarely happen that one string will be a *perfect* substring of another. Rather, it is often necessary to introduce "gaps" in one string or the other in order to get a good match. Moreover, one often has to settle for "matches" of one symbol in one string against a different (i.e., not identical) symbol in the other string. To formulate this problem precisely, define a "weight" matrix $w : \mathcal{A} \times \mathcal{A} \to \Re$. Thus if a symbol $x \in \mathcal{A}$ in one string is aligned against a symbol $y \in \mathcal{A}$ in the other string, the weight assigned to the match is $w(x, y)$. Typically $w(x, x)$ is large and positive, while $w(x, y)$ is negative (and either large or small) if $x \neq y$. In case a gap is introduced in one or the other string, one can assign a gap penalty. Note that it makes no sense to place gaps in both strings, one against the other. Thus if a gap is introduced in one string and then extended, the weight can be twice the gap penalty; alternatively, the penalty for *introducing* a gap can be larger than the incremental penalty for *extending* a gap. In any case, the sum of all the weights from one end to the other is the total score of the alignment. The optimal gapped alignment is one that minimizes the total weight. Note that, in any gapped alignment (optimal or otherwise) the total length of the gapped strings is always equal, whether or not the original strings are of equal length.

The optimal gapped alignment problem can be readily solved using dynamic programming, for the simple reason that an optimal gapped alignment satisfies the *principle of optimality*, that is: a sub-alignment of an optimal alignment is itself optimal. A dynamic programming solution to the optimal gapped alignment problem is called the Needleman-Wunsch algorithm. If n denotes the length of the larger string $\mathbf{v}$, then the computational complexity of this algorithm is $O(n^2)$. By a slight modification, the dynamic programming aproach can also be used to find an optimal gapped alignment between *substrings* of $\mathbf{u}$ and $\mathbf{v}$.

There are several problems that are still unsolved. While the dynamic programming approach has *polynomial* complexity in terms of the *length* of the strings, if one wishes to align more than two strings, then the complexit is *exponential* in terms of the *number* of strings. On the other hand, in biology it is really not necessary to find an *optimal* alignment. It is only necessary to find some kind of resemblance between strings. Thus it would be worthwhile to find some alignment algorithms that are suboptimal but in a guaranteed sense. In the same way, it appears to the author that the use of *randomized algorithms* should be explored.

See [5, 6] for a discussion of string alignment algorithms. Two of the papers in this special session (by Aluru and by Wise) address the problem of string alignment.

4 Markov Chains and Hidden Markov Models

In this section we briefly review some standard material on Markov chains. Then the discussion is extended to so-called hidden Markov models (HMM's). Markov models are discussed in many standard texts, such as [6, 7] and so on. Hidden Markov models (HMM's) are used to separate the coding regions of a Prokaryote gene from the non-coding regions, and also to classify a protein into one of a small number of previously classified protein families.

4.1 Markov Chains

Suppose $X := \{s_1, \ldots, s_n\}$ is a finite set. A stochastic process $\{\mathcal{X}_t\}_{t \geq 0}$ assuming values in X is said to be a **Markov chain** if

$$\Pr\{\mathcal{X}_t | \mathcal{X}_{t-1}, \mathcal{X}_{t-2}, \ldots\} = \Pr\{\mathcal{X}_t | \mathcal{X}_{t-1}\}.$$

The Markov chain is said to be **stationary** if the above conditional probability is independent of t. The temporal evolution of a Markov chain is captured by an $n \times n$ matrix of transition probabilities, defined as follows:

$$a_{ij} := \Pr\{\mathcal{X}_t = s_j | \mathcal{X}_{t-1} = s_i\}, \ A = [a_{ij}].$$

Thus a_{ij} is the probability that the Markov chain is in state s_j at the next time instant, given that it is in the state s_i at the current time instant. Obviously the matrix A is row-stochastic; that is,

$$a_{ij} \geq 0 \ \forall i, j, \text{ and } \sum_{j=1}^{n} a_{ij} = 1 \ \forall i.$$

Hence the vector all one's is a *column* eigenvector of A. A standard result (see e.g., [3], Theorem 3.11) states that every such matrix also has at least one *row* eigenvector whose components are all nonnegative (and can therefore be scaled to add up to 1). Such a vector, i.e., a nonnegative vector π such that $\pi = A\pi$ and such that the components of π add up to 1, is called a *stationary* distribution. Under some additional conditions, such as the irreducibility of the Markov chain, there is only one stationary distribution. Note tha if the Markov chain is started off at time $t = 0$ with the initial state $\mathcal{X}_0$ distributed according to π, then $\mathcal{X}_t$ is distributed according to π at all future times.

4.2 Hidden Markov Models

One of the main motivations for studying Markov chains is that in some sense they have a finite description. The Markov property says that the latest measurement $\mathcal{X}_{t-1}$ contains all the information contained in *all* the past measurements. Thus, once the observer measures $\mathcal{X}_{t-1}$, he can throw all past measurements without any loss of information. Now what happens if a stochastic process is not Markovian? Hidden Markov models (HMM's) are somewhat more general models for stochastic processes that still retain the "finite memory" feature of Markov chains.

Specifically, suppose we now have *two* sets $X := \{s_1, \ldots, s_n\}$ called the state space, and $Y := \{r_1, \ldots, r_m\}$ called the output

space. Suppose $\{\mathcal{X}_t\}$ is a Markov chain. At each time t, the state $\mathcal{X}_t$ induces a probability distribution on the output space Y, as follows:

$$\Pr\{\mathcal{Y}_t = r_i | \mathcal{X}_t = s_j\} = b_{ij}, \ \forall i, j.$$

The stochastic process $\{\mathcal{Y}_t\}$ is said to obey a hidden Markov model (HMM). Thus a HMM is described by an $n \times n$ matrix A of state transition probabilities, and another $m \times n$ matrix B of readout probabilities.

Suppose we know the matrices A, B, and we have a single sample path $\{y_t\}_{t\geq 0}^T$ of observations of the stochastic process $\{\mathcal{Y}_t\}$. At this point, one can ask three distinct questions.

1. Given the matrices A, B, what is the likelihood of this particular sample path? This is called the "likelihood question" and requires us to compute the quantity

$$\Pr\{\mathcal{Y}_t = y_t \ \forall t\}, \text{ given } A, B.$$

2. What is the most likely sequence of states $\{x_t\}_{t\geq 0}^T$? This is called the "decoding question."

3. Assuming that we know only the integer n but not the entries in the matrices A and B, can we iteratively compute these entries based on an observation of a sample path $\{y_t\}_{t\geq 0}^T$? This is called the "learning question."

Answers to these questions have been known for many years; see [2], Chapter 7 for a discussion of these issues.

But there is another, more fundamental question that is addressed in a companion paper by this author, namely: Suppose we have a stationary stochastic process $\{\mathcal{Y}_t\}$ assuming values in a finite set Y. Do there exist an integer n, a set X of cardinality n, and matrices A, B such that $\{\mathcal{Y}_t\}$ is precisely the corresponding HMM? A partial answer is given in [1] and a complete answer is given in the companion paper.

5 Classification of Proteins into Families Using HMM's

One of the successful applications of HMM's is in classifying a new protein into one of a small number (typically three or four) of protein families; see [10] for details.

In simplified form the problem is as follows: One begins with a fairly large number of proteins, which are grouped into a small number of families on the basis of structural or functional similarities. Typical numbers are: 500 to 1,000 for the total number of proteins, and three to four for the number of families. Then a new protein is given in terms of its primary structure, and the objective is to determine to which of the small number of protein families the new protein is most likely to belong.

To solve this problem using HMM's, one proceeds as follows. For each of the families, one develops a corresponding HMM. First, to cater for the fact that the primary structures of the various proteins within a particular family all have different lengths, one carries out a gapped alignment of the protein sequences. Note that this gapped alignment is usually done "by hand" since it would be computationally infeasible to carry out an *optimal* gapped alignment. [1] Once the multiple alignment is carried out, one has a large number of amino acid sequences, *now all of equal length*. Now one views all these gapped aligned sequences as sample paths (of equal length) of the output of a single HMM. To take into account the gaps in the alignment, the output space is taken as having a cardinality of 21 (the 20 amino acids plus the gap symbol).

Figure 1: Hidden Markov Model for Protein Classification

Figure 1 above shows the HMM used to produce each of these sample paths. At each step along the aligned sequences (which are now viewed as evolving *in time*, rather than along spatial ordering), one computes the frequency of each of the three events occuring, namely: deletion of an acid, insertion of an acid, or mutation of one acid into another. In each of these states, the 21-dimensional probability vector corresponding to each of the 21 possible outputs is computed as just the actual frequency of occurence of these 21 outputs in the observed sequences. This completes the specification of the HMM for a particular family. Similar constructions are done for each of the small number of protein families.

To classify a new protein sequence, one first does a multiple gapped alignment with all the proteins within each family. This generates a sample path corresponding to each of the three or four HMM's. Then the likelihood of the sample path is computed for each HMM. The HMM for which the sample path likelihood is maximum is declared as the winner, i.e., the new protein sequence is classified as belonging to that particular family.

In a companion paper, it is pointed out that the above method has a number of drawbacks. For one thing, the underlying Markov chain is *reducible*. This is because there is no path from a state at time i to a state at time $j < i$. In reality, this reducibility comes about because the HMM is actually trying to simulate a *non-stationary* stochastic process as a stationary process. The second thing to notice about the HMM is that it has a *huge* number of parameters to be estimated. Both of these issues are addressed in a companion paper.

[1] Recall that the complexity of the existing optimal gapped alignment algorithms is *exponential* in the number of strings being aligned.

6 Estimating the Transition Probabilities of a Markov Chain

We conclude the paper by raising an important problem for which the literature contains only a partial (and somewhat unsatisfactory) solution. Suppose we have a Markov chain evolving over a known finite state space $X = \{1, \ldots, n\}$, but with an unknown state transition matrix A. How is it possible to estimate the entries of A based on observing a realization of the Markov chain?

Let a_{ij} denote the ij-th entry of A. After watching N time steps of the Markov chain, we can form an estimate of $\hat{\mathbf{a}}_{ij}$ for a_{ij} by defining

$$\hat{\mathbf{a}}_{ij} = \frac{\sum_{t=1}^{N} I\{X_{t+1} = j \& X_t = i\}}{\sum_{t=1}^{N} I\{X_t = i\}},$$

where $I\{\cdot\}$ equals one if the event occurs and equals zero otherwise. Thus $\hat{\mathbf{a}}_{ij}$ is the fraction of times that, starting in state i at time t, the Markov chain makes a transition to the state j. Since successive cycles of a Markov chain are independent, it is possible to use Hoeffding's inequality to conclude that

$$\Pr\{|\hat{\mathbf{a}}_{ij} - a_{ij}| > \epsilon\} \leq 2\exp(-2N_i\epsilon^2),$$

where N_i is the number of times that the Markov chain passes through state i during the first N observations.

The main difficulty with the above bound is that N_i is itself a random variable. If the Markov chain starts off in the (unknown) stationary distribution π, then $N_i \approx \pi_i N$. However, if some components of π are very small, then we need to make a very large number of observations in order for the error $|\hat{\mathbf{a}}_i - a_{ij}|$ to be *uniformly* small with respect to i and j. This in turn requires us to make some assumptions about the magnitude of the smallest component of π. In other words, our estimates of the rate of convergence of the estimates to the true values depend on the unknown quantities – a circular reasoning. It is highly desirable to find ways around this difficulty.

7 Discussion

In this paper, we have discussed a few problems in computational biology that are amenable to analysis using a "systems" viewpoint. The references, especially [2, 6], contain several other such problems.

References

[1] B. D. O. Anderson, "The realization problem for hidden Markov models," *Math. Control, Sig., Sys.*, 12(1), 80-120, 1999.

[2] P. Baldi and S. Brunak, *Bioinformatics: A Machine Learning Approach*, MIT Press, Cambridge, MA, 2001.

[3] A. Berman and R. J. Plemmons, *Nonnegative Matrices*, Academic Press, New York, 1979; reprinted in the series *Classics in Applied Mathematics*, No. , SIAM Publications, Philadelphia, PA, .

[4] A. Dembo, S. Karlin and O. Zeitouni, "Limit distributions of maximal non-aligned two-sequence segmental score," *Ann. Prob.*, 22(4), 2022-2039, 1994.

[5] R. Durbin, S. Eddy, A. Krogh and G. Mitchison, *Biological Sequence Analysis*, Cambridge University Press, Cambridge, UK, 1998.

[6] W. J. Ewens and G. R. Grant, *Statistical Methods in Bioinformatics*, Springer-Verlag, New York, 2001.

[7] S. Karlin, *A First Course in Stochastic Processes*, Academic Press, New York, 1969.

[8] S. Karlin and S. F. Altschul, "Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes," *Proc. Nat'l. Acad. Sci. (USA)*, 87, 2264-2268, 1990.

[9] S. Karlin and A. Dembo, "Limit distributions of maximal segmental score among Markov-dependent partial sums," *Adv. Appl. Prob.*, 24, 113-140, 1992.

[10] A. Krogh, M. Brown, I. S. Mian, K. Sjölander and D. Haussler, "Hidden Markov models in computational biology: Applications to protein modeling," *J. Mol. Biol.*, 235, 1501-1531, 1994.