Chapter 3

STATISTICAL METHODS IN BIO-INFORMATICS

3.1 Introduction

The definition of Bioinformatics was outlined in chapter-1, however, is reiterated here for the sake of continuity to understand the significance of statistical methods. In the last chapter, we have outlined the applications of clustering in various fields, which is one of the important statistical tools used most in Bioinformatics. In addition, there are large set of statistical methods made use of in Bioinformatics and some of them are outlined which are contemplated as future academic agenda by the scholar. Bioinformatics is the science of managing, analyzing, extracting, and interpreting information from biological sequences and molecules. It has been an active research area since late 1980’s. After the human genome project was completed in April 2003, this area has drawn even more attention of Computer Scientists, Statisticians, Experts of Machine learning and all other allied Scientists. With more genome sequencing projects under taken, data in the field such as DNA sequences, protein sequences and protein structures are exponentially growing. Facing this enormous amount of data in terms of terabytes, the biologists failed to use traditional techniques in Biology to analyze the data but had to depend on information technologies to understand several mysterious aspects of life which lead to applications of data mining, to extricate the biological information of all living organisms. Generally, Data mining techniques deal with three major issues of classification, clustering, and association.

Clustering and Classification methods have many applications in Bioinformatics such as protein homology detection, protein family classification, metabolic pathway clustering etc. In the above applications hierarchical correlations are to be tackled in an effective and efficient way with the incorporation of domain specific knowledge. With this brief introduction we present various statistical methods or tools required in the analysis of biological data. In this work, we briefly state the importance of some of the statistical methods below, 1.Bayesian modeling, 2.Probabilistic modeling, 3.Dynamic Programming, 4.Markov Chain Monte Carlo methods, 5.Artificial Neural Networks,

3.2 Bayesian Modeling

Computational molecular biologists are, then, constantly faced with induction and inference problems - building models from available data. What are the right class of models and the right level of complexity? Which details are important and which should be discarded? How can one compare different models and select the best one, in light of available knowledge and sometimes limited data? In short, how do we know whether a particular model is a good fit? These questions are all very essential in machine learning approaches (for the development of software for expert systems), because complex models, with several thousand parameters and, are routinely used in sequencing abundant data having inherent noise.

When reasoning in the presence of certainty, one uses deduction. This is how the most advanced theories in information-poor sciences, such as physics or mathematics, are presented in an axiomatic fashion. Deduction is not controversial. The overwhelming majority of people agree on how to perform deductions in a specific way: if $X$ implies $Y$, and $X$ is true, then $Y$ must be true. This is the essence of Boole’s algebra, and at the root of all our digital computers. When reasoning is in the presence of uncertainty, one uses induction and inference: if $X$ implies $Y$, and $Y$ is true, then $X$ is more plausible. An amazing and still poorly known fact is that there is a simple and unique consistent set of rules for induction, model selection, and comparison. It is called Bayesian inference. More precisely, in order to carry out the induction process, one ought to proceed as follows:

1. Clearly state what the hypotheses or models are, along with all the background information and the data.
2. Use the language of probability theory to assign prior probabilities to the hypotheses.
3. Use probability calculus for the inference process, in particular to evaluate posterior probabilities for the hypotheses in the light of the available data, and to derive unique answers.
The hierarchical structure in metabolic pathway, protein homology etc. can be classified using a tree structural dendrograms. Once such interdependent tree structure is identified the general probabilistic framework can be achieved through graphical models to factor high dimensional probability distributions by exploiting independent assumptions that have a graphical substrate, can be represented in terms of recursive sparse graphs at the levels of both variables involved, observed or hidden, and the parameters. Analysis of sparse recursive graphs can be achieved through Bayesian inference.

3.3 Probabilistic Modeling

The simplest, but not entirely trivial, modeling situation is that of a single coin flip. This model has a single parameter p and the data consist of a string, containing a single letter, over the alphabet \( A = \{H, T\} \), H for head and T for tail. Since we are interested in DNA sequences, we shall move to a slightly more complex version with four letters, rather than two, and the possibility of observing longer sequences.

The data \( D \) then consist of DNA strings over the four letter alphabet \( A = \{A, C, G, T\} \). The simple model we want to use is to assume that the strings have been obtained by the independent tosses of the same four sided die. Because the tosses are independent and there is a unique underlying die, for likelihood considerations it does not really matter whether we have many strings or a single long string. So we assume that the data consist of a single observation sequence of length \( N \): \( D = \{O\} \), with \( O = \{X^1 \ldots X^N\} \) and \( X^i \in A \).

Then the model \( M \) has four parameters \( p_A, p_C, p_G, p_T \) satisfying \( p_A + p_C + p_G + p_T = 1 \). The likelihood is then given by

\[
P(D|M) = \prod_{x \in A} p_x^{n_x} = p_A^{n_A} p_C^{n_C} p_G^{n_G} p_T^{n_T}
\]  

(3.1)

where \( n_x \) is the number of times the letter \( x \) appears in the sequence \( O \). The negative log-posterior is then

\[
-\log P(M|D) = -\sum_{x \in A} n_x \log p_x - \log P(M) + \log P(D).
\]  

(3.2)

If we assume a uniform prior distribution over the parameters, then the Maximum a Posteriori (MAP) parameter estimation problem is identical to the ML parameter estimation problem and can be solved by optimizing the Lagrangian

\[
\mathcal{L} = -\sum_{x \in A} n_x \log p_x - \lambda (1 - \sum_{x \in A} p_x)
\]  

(3.3)
associated with the negative log-likelihood and augmented by the normalizing constraint.

### 3.4 Dynamic Programming

Dynamic programming (Bertsekas 1995) is to a very general optimization technique that can be applied any time a problem can be recursively subdivided into two similar sub problems of smaller size, such that the solution to the larger problem can be obtained by piecing together the solutions to the two sub problems. The prototypical problem to which dynamic programming can be applied is that of finding the shortest path between two nodes in a graph. Clearly the shortest path from node $A$ to node $B$, going through node $C$, is the concatenation of the shortest path from $A$ to $C$ with the shortest path from $C$ to $B$. This is also called the “Bellman principle.” A general solution to the original problem is then constructed by recursively piecing together shorter optimal paths. Dynamic programming and its many variations are ubiquitous in sequence analysis. The Needleman–Wunch and Smith–Waterman algorithms (Needleman SB and Wunsch 1970, Sellers 1974, Smith and Waterman 1981), as well as all other sequence-alignment algorithms such as the Viterbi decoding algorithm of electrical engineers, are examples of dynamic programming. Alignment algorithms can be visualized in terms of finding the shortest path in the appropriate graph with the appropriate metric. Aligning two sequences of length of $N$ requires finding a shortest path in a graph with $N^2$ vertices. Since dynamic programming essentially requires visiting all such vertices once, it is easy to see that its time complexity scales as $O(N^2)$.

**Markov - Chain Monte - Carlo Methods**

Markov-chain Monte-Carlo (MCMC) methods belong to an important class of stochastic methods that are related to statistical physics and are increasingly used in Bayesian inference and machine learning (York 1992, Geyer 1992, Neal 1993, Tierney 1994, Besag et al. 1995). Recall that one of the basic goals derived from the general Bayesian framework is to compute expectations with respect to a high-dimensional probability distribution $P(x_1, \ldots, x_n)$, where the $x_i$ can be the values of model parameters or hidden variables, as well as observed data. The two basic ideas behind MCMC are very simple. The first idea (Monte Carlo) is to approximate such expectations by

$$
E(f) = \sum_{x_1, \ldots, x_n} f(x_1, \ldots, x_n) P(x_1, \ldots, x_n) = \frac{1}{T} \sum_{t=0}^T f(x'_1, \ldots, x'_n)
$$

(3.4)
for large $T$, provided $(x'_1, \ldots, x'_n)$ are sampled according to their distribution $P(x_1, \ldots, x_n)$.

In order to sample from $P$, the second basic idea is to construct a Markov chain having $P$ as its equilibrium distribution, then simulate the chain and try to sample from its equilibrium distribution.

**Markov Chains**

Markov chains with finite state space always have at least one stable distribution. Obviously, in MCMC sampling procedures, we will be interested in stable distributions, in fact in the even stronger conditions of *ergodic* distributions. Here, a distribution is defined to be ergodic if and only if the chain always converges to it, regardless of the choice of the initial distribution at time 0. In the case of an ergodic Markov chain, there is only one stable distribution, called the *equilibrium* distribution. Conditions for the ergodicity of a Markov chain, and bounds on the rate of convergence to the equilibrium distribution, are well known (Diaconis and Stroock 1991, Fill 1991). In order to achieve our goal of sampling from $P(x_1, \ldots, x_n)$, we now turn to the two main MCMC algorithms: Gibbs sampling and the Metropolis algorithm.

**3.5 Artificial Neural Networks**

Artificial neural networks (ANNs) (Rumelhart *et al.* 1986, Hertz *et al.* 1991, Bishop 1995) were originally developed with the goal of modeling information processing and learning in the brain. While the brain metaphor remains a useful source of inspiration, it is clear today that the artificial neurons used in most NNs are quite remote from biological neurons (Bower and Beeman 1995). The development of NNs, however, has led to a number of practical applications in various fields, including computational molecular biology. NNs have become an important tool in the arsenal of machine-learning techniques that can be applied to sequence analysis and pattern recognition problems. At the most basic level, NNs can be viewed as a broad class of parameterized graphical models consisting of networks with interconnected units evolving in time. In this work we use only pairwise connections but, if desirable, one can use more elaborate connections associated with the interaction of more than two units, leading to the “higher-order” or “sigma-pi” networks. A number of architectures such as recurrent, feed-forward, and layered depending on the type of modeling used in a given application. One important issue, before we can proceed with NN applications to molecular biology, is the
encoding of the sequence input. In any type of prediction approach, the input representation is of cardinal importance. If a very clever input representation is chosen, one that reveals exactly the essentials for a particular task, the problem may be more or less solved, or at least can be solved by simple linear methods. In a Multilayer Perceptron (MLP) the activity patterns in the last hidden layer preceding the output unit(s) should represent the transformed input information in linearly separable form. This clearly is much easier if the input representation has not been selected so as further to increase the nonlinearity of the problem. One would think that a very “realistic” encoding of the monomers in a sequence, using a set of physical-chemical features of potential relevance, should always outperform a more abstract encoding taken from the principles and practice of information theory (Cover and Thomas 1991). However, in line with the contractive nature of most prediction problems, it does not always help just to add extra information because the network has to discard most of it before it reaches the output level. So like this many problems in computational molecular biology with nonlinearity nature can be easily solved with the tool of Artificial Neural Networks.

### 3.6 Hidden Markov models

In the 1990s, only roughly a third of the newly predicted protein sequences show convincing similarity to other known sequences (Bork et al. 1992, Green et al. 1993, Eddy 1996), using pairwise comparisons (Altschul et al. 1990, Pearson and Lipman 1988). This situation is even more unfortunate in the case of new, incomplete sequences or fragments. Large databases of fragments are becoming available as a result of various genome, cDNA, and other sequencing projects, especially those producing ESTs (expressed sequences tags) (Gerhold and Caskey 1996). At the beginning of 1997, approximately half of GenBank consisted of fragment data. Such data cover a substantial fraction, if not all, of the expressed human genome. It is of course of great interest to recognize and classify such fragments, and recover any additional useful information.

HMMs form another useful class of probabilistic graphical models used, over the past few decades, to model a variety of time series, especially in speech recognition (Levinson et al. 1983, Rabiner 1989) but also in a number of other areas, such as ion channel recordings and optical character recognition. HMMs have also earlier been applied to problems in computational biology, including the modeling of coding/noncoding regions...
in DNA (Churchill 1989), of protein binding sites in DNA (Lawrence and Reilly 1990), and of protein super families.

Regardless of the design and training method, once an HMM has been successfully derived from a family of sequences, it can be used in a number of different tasks, including

- Multiple alignments
- Data mining and classification of sequences and fragments
- Structural analysis and pattern recognition

### 3.7 Probabilistic Models of Evolution for Phylogenetic Trees

Little work has been done so far to define prior distributions on the space of phylogenetic trees, in terms of both the branching process and the branching lengths. For a given topology, the lengths can be viewed as the parameters of the models and therefore can be optimized by Maximum Likelihood (ML). As for HMMs, in general the ML estimate cannot be determined analytically but can be approximated using, for example, gradient descent, Expectation Maximization (EM), or perhaps some form of Viterbi learning.

The optimization of the topology is a second problem that requires approximations. The number of possible trees, even unrooted, is exponentially large and the space of topologies cannot be searched exhaustively. One widely used heuristic algorithm consists of progressively adding species (i.e., observation sequences) one by one, starting with a two-species tree. At each step, a new species is selected and all its possible positions with respect to the current tree are considered. The most likely is selected before proceeding to the next step. One serious caveat with a search algorithm of this sort is that the final tree topology depends on the order of presentation of the observation sequences. In applying ML approach to phylogenetic trees is rather computationally intensive. A complete Bayesian treatment of phylogeny is even more intensive since, in addition to priors, it requires integrating across trees in order, for instance, to estimate the probability that a given substitution has or has not occurred in the past. Parsimony methods can be viewed as fast approximations to ML. The basic idea behind parsimony is that the optimal tree is the one requiring the smallest number of substitutions along its branches. In this sense, it is somewhat related to MDL (minimum description length) ideas.
For applying parsimony techniques the computer scientist requires thorough knowledge of computational aspects of sparse matrices, hence poses challenges in the representation of data structures.

### 3.8 Stochastic Grammars and Linguistics

Formal grammars were originally developed to model natural languages, around the same time that the double-helical structure of DNA was elucidated by Watson and Crick. Since then, grammars have been used extensively in the analysis and design of computer languages and compilers (Aho et al. 1986). Grammars are natural tools for modeling strings of letters hence are applied to biological sequences. The theory of formal languages, including different classes of grammars, their properties, the Chomsky hierarchy, and the connection to HMMs can be applied to biological sequences and especially the application of context free grammars to RNA molecules. Ultimately, one would like to derive grammar models all the way up to the scale of genes, chromosomes, and even genomes. After all, genomes represent only a very small fraction of all the possible DNA sequences of comparable length. And therefore we find several applications of stochastic grammars in modeling RNA secondary structure, biological palindromes, and extending it to applications of context free grammars for RNA. One can go beyond the realm of context free grammar, push down automata and nested dependencies. After modeling the grammars, another challenging task is the Bayesian estimations that is defining Dirichlet priors, likelihood functions, learning algorithms such as Expectation maximization, Gradient descent, viterbi learning etc.