BIOLOGICAL MODELING AND SIMULATION

A Survey of Practical Models, Algorithms, and Numerical Methods

Russell Schwartz

The MIT Press
Cambridge, Massachusetts
London, England

This book was set in Times New Roman and Syntax on 3B2 by Asco Typesetters, Hong Kong. Printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Schwartz, Russell.
Biological modeling and simulation : a survey of practical models, algorithms, and numerical methods / Russell Schwartz.
p. cm. — (Computational molecular biology)
Includes bibliographical references and index.
ISBN 978-0-262-19584-3 (hardcover : alk. paper) 1. Biology—Simulation methods. 2. Biology—Mathematical models. I. Title.
QH323.5.S364 2008
570.1′1—dc22 2008005539

10 9 8 7 6 5 4 3 2 1

1 Introduction

1.1 Overview of Topics

This book is divided into three major sections: models for optimization, simulation and sampling, and parameter-tuning. Though there is some overlap among these topics, they provide a general framework for learning how one can formulate models of biological systems, what techniques one has to work with those models, and how to fit those models to particular systems.

The first section covers perhaps the most basic use of mathematical models in biological research: formulating optimization problems for biological systems. Examples of models used for optimization problems include the molecular evolution models generally used to formulate sequence alignment or evolutionary tree inference problems, energy functions used to predict docking between molecules, and models of the relationships between gene expression levels used to infer genetic regulatory networks. We will start with this topic because it is a good way for those who already have some computational background to get experience in reasoning about how to formulate new models.

The second section covers simulation and sampling (i.e., how to select among possible system states or trajectories implied by a given model). Examples of simulation and sampling questions we could ask are how a biochemical reaction system might change over time from a given set of initial conditions, how a population might evolve from a set of founder individuals, and how a genetic regulatory network might respond to some outside stimulus. Answering such questions is one of the main functions of models of biological systems, and this topic therefore takes up the greatest part of the text.

The third section covers techniques for fitting model parameters to experimental data. Given a data set and a class of models, the goal will be to find the best model from the class to fit the data. A typical parameter-tuning problem would be to estimate the interaction energy between any two amino acids in a protein structure model by examining known protein structures. Parameter-tuning overlaps with

optimization, as finding the best-fit parameters for a model is often accomplished by optimizing for some quality metric. There are, however, many specialized optimization methods that frequently recur in parameter-tuning contexts. We will conclude our discussion of parameter-tuning by considering how to evaluate the quality of whatever fit we achieve.

1.2 Examples of Problems in Biological Modeling

To illustrate the nature of each of these topics, we can work through a few simple examples of questions in biology that we might address through computational models. In this process, we can see some of the issues that come up in reasoning about a model.

1.2.1 Optimization

Often, when we examine a biological system, we have a single question we want to answer. A mathematical model provides a way to precisely judge the quality of possible solutions and formulate a method for solving it. For example, suppose I have a hypothetical group of organisms: a bacterium, a protozoan, a yeast, a plant, an invertebrate, and a vertebrate. Our question is "What are the evolutionary relationships among these organisms?" That may seem like a pretty straightforward question, but it hides a lot of ambiguity. By modeling the problem, we can be precise about what we are asking.

The first thing we need is a model of what "evolutionary relationships" look like. We can use a standard model, the evolutionary tree. Figure 1.1 shows a hypothetical (and rather implausible) example of an evolutionary tree for our organisms. Note that by choosing a tree model, we are already restricting the possible answers to our question. The tree leaves out many details that may be of interest to us, for example, which genes are conserved among subsets of these organisms. It also makes assumptions, such as a lack of horizontal transfer of genes between species, that may be inaccurate when understanding the evolution of these organisms. Nonetheless, we have to make some assumptions to specify precisely what our output looks like, and these

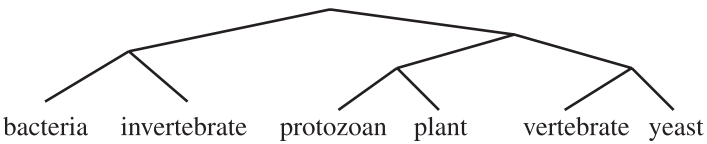

Figure 1.1
Hypothetical evolutionary tree linking our example set of organisms.

are probably reasonable ones. We have now completed one step of formalizing our problem: specifying our *output format*.

We then must deal with another problem: even if our model specifies that our output is a tree, we do not know which one. We cannot answer our question with certainty, so what we really want to find is the best answer, given the evidence available to us. So, what is the evidence available to us? We might suppose that our evidence consists of genetic sequences of some highly conserved gene or genetic region in each organism. That means we assume we are given m strings on an alphabet $\{A, C, T, G\}$. Figure 1.2 is an example of such strings that have been aligned to each other by inserting a gap ("-") in one. We have now completed another step in formalizing our problem: specifying our *input format*.

Now we face another problem. There are many possible outputs consistent with any input. So which is the best one? To answer that, our model needs to include some measure of how well any given tree matches the data. A common way to this is to assume some model of the process by which the input data may have been generated by the process of evolution. This model will then have implications for the probability of observing any given tree. Let us propose some assumptions that will let us define a formal model:

- Our gene is modified only by point mutations, changing one base at a time.
- Mutations are rare.
- Any one mutation (or insertion or deletion) is as likely to occur as any other.
- Mutations are selectively neutral, that is, they do not affect the probability of the organism's surviving and reproducing.

Those are not exactly correct assumptions, but they may be reasonable approximations, depending on the characteristics of our problem. Given these assumptions, we might propose that the best tree is the one that involves the fewest mutations between organisms. A model that seeks to minimize some measure of complexity of the solution is called a *parsimony* model. Parsimony formulations often lead to recognizable optimization problems. In this case, we can define an *edit distance* d between

```
ACGGTAC
ACCGAAC
AC-GGAC
   .
   .
   .
```

Figure 1.2
A set of strings on the alphabet $\{A, C, T, G\}$ that have been aligned to each other.

two strings s_1 and s_2 to be the minimum number of insertions, deletions, and base changes necessary to convert one string into the other. Then our solution to the problem will consist of a tree with leaves labeled with our input strings and with internal nodes labeled with other strings such that the sum of the edit distances across all edges in the tree is minimized. We have now accomplished the third task in formalizing our problem: specifying a *metric* for it.

Now that we have the three components of our formal specification—an *input format*, an *output format*, and a *metric*—we have specified our model well enough to formulate a well-defined computational optimization problem. We can take the same problem we specified informally above and write it more formally as follows:

Input A set S of strings on the alphabet $\Sigma = \{\mathrm{A}, \mathrm{C}, \mathrm{T}, \mathrm{G}\}$ representing our DNA sequences to be examined
Output A tree $T = (V, E)$ with $|S|$ leaves $L \subseteq V$ and an assignment of string tags to nodes $t : V \rightarrow \Sigma^*$ satisfying the constraint $\forall s \in S \exists l \in L$ s.t. $t(l) = s$ (read as "for all strings s in set S, there exists a leaf node l from set L such that the tag of l, $t(l)$, is the string s")
Metric $\sum_{(u,v) \in E} d(t(u), t(v))$ (read as "the sum over all edges u to v in the edge set E of the edit distance between the tag of u, $t(u)$ and the tag of v, $t(v)$") is minimized over trees T and tag assignments t.

In other words, we want to find the tree whose leaves are labeled with the sequences of our organisms and whose internal nodes are labeled with the sequences of presumed common ancestors such that we minimize the total number of base changes over all pairs of sequences sharing an edge in the tree. This does not yet tell us how to solve the problem, but it does at least tell us what problem to solve. Later in the book, we will see how we might go about solving that problem.

1.2.2 Simulation and Sampling

Another major use of models is for simulation. Usually, we use simulations when we are interested in a process rather than a single outcome. Simulating the process can be useful as a validation of a model or a comparison of two different models. If we have reason to trust our model, then simulation can further be used to explore how interventions in the model might affect its behavior. Simulations are also useful if the long-term behavior of the model is hard to analyze by first principles. In such cases, we can look at how a model evolves and watch for particularly interesting but unexpected properties.

As an example of what one might do with simulation, let us consider an issue motivated by protein structure analysis. Suppose we are given the structure of a protein and we wish to understand whether we can mutate the protein in some way that increases its stability. Simulations can provide a way to answer this sort of question.

Our input can be assumed to be a protein sequence (i.e., a string of amino acids). More formally, our input is a string $s \in \Sigma^*$ ("Σ^*" is a formal notation for a string of zero or more characters from the alphabet Σ), where $\Sigma = \{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}$.

If we want to answer this question, we first need a model for the structure of our protein. For the purposes of this illustration, we will use a common form of simplified model called a lattice model. In a lattice model, we treat a protein as a chain of beads sitting at points on a regular grid. To simplify the illustration, we will represent this as a two-dimensional structure sitting on a square grid. In practice, much more flexible lattices are available that better capture the true range of motion of a protein backbone. Lattice models tend to be a good choice for simulations involving protein folding because they are simple enough to allow nontrivial rearrangements to occur on a reasonable time scale. They are also often used in optimizations related to protein folding because of the possibility of enumerating discrete sets of conformations in them. Our model of the protein structure is, then, a self-avoiding chain on a 2-D square lattice (see figure 1.3).

If we want to study protein energetics, we need a model of the energy of any particular structure. Lattice models are commonly used with *contact potentials* that assign a particular energy to any two amino acids that are adjacent on the lattice but not in the protein chain. For example, in the model protein above, we have two contacts, S to L at the top and D to K at the bottom. These are shown as thick dashed lines in figure 1.3. On more sophisticated lattices, these potentials might vary with distance between the amino acids or their orientations relative to one another, but we will ignore that here.

As a first pass at solving our problem, we might simply stop here and say that we can estimate the stability effect of an amino acid change by looking at the change in contact energies it produces. For example, suppose our model specifies a contact energy of +1 kcal/mol for contact between S and L and −1 kcal/mol for contact

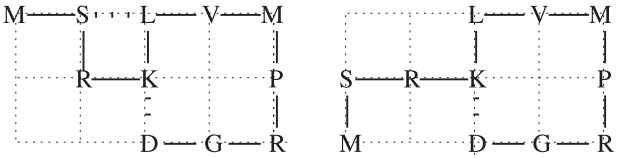

Figure 1.3
A hypothetical protein folded on a lattice. Solid lines represent the path of the peptide backbone. Thick dashed lines show contacts between amino acids adjacent on the lattice but not on the backbone. Thin dashed lines show the lattice grid. (a) Initial conformation of the protein. (b) Alternative conformation produced by pivoting around the arginine (R) amino acid.

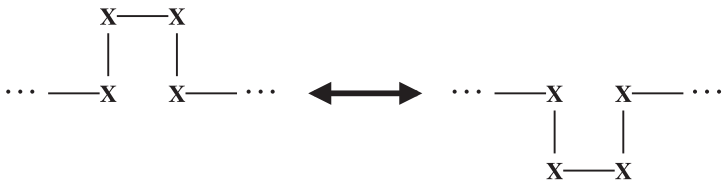

Figure 1.4
An example of a lattice move. X stands for any possible amino acid, and the ellipses stand for any possible conformation of the chain outside of a local region of interest. This move indicates that a 180° bend of four residues can be flipped about the surrounding backbone.

between S and T. Then we might propose that if the conformation in figure 1.3(a) is our protein's native (normal) state, then mutating L to T will increase stability (reduce energy) by 2 kcal/mol. We might then propose to solve the problem by attempting substitutions at all positions until we find the set of amino acids with the lowest possible energy summed over all contacts. This first-pass solution is problematic, though, in that it neglects the fact that an amino acid change which stabilizes the native conformation might also stabilize nonnative conformations. The change might thereby reduce the time spent in the native state even while reducing the native state's intrinsic energy.

We therefore need some way to study how the protein might move under the control of our energy model. There are many *move sets* for various lattices that attempt to capture how a protein chain might bend. A move set is a way of specifying how any given conformation can be transformed into other conformations. Figure 1.4 shows an example of a possible move for a move set. Anywhere we observe a subset of a conformation matching the left pattern, it would be legal to transform it to match the right pattern, and vice versa. This move alone would be insufficient to create a realistic folding model, but it might be part of a larger set allowing more freedom of movement. For this small example, though, we will assume a simpler move set. We will say that a single move of a protein consists of choosing any one bond in the protein and bending it to any arbitrary position that does not produce collisions in the chain. We can get from any chain configuration to any other by some sequence of these single-bond bends. For example, we could legally change our chain configuration in figure 1.3(a) into that in figure 1.3(b) by pivoting 90° at the S-R-K bend. We would not be able to pivot an additional 90°, though, because that would create a collision between the M and D amino acids.

The move set only tells us which moves are allowed, though, not which are likely. We further need a model of *dynamics* that specifies how we select among different legal moves at each point in time. One common method is the *Metropolis criterion*:

1. Pick uniformly at random among all possible moves from the current conformation C_1 to some neighboring conformation C_2.
2. If the energy of C_2 is less than the energy of C_1, *accept* the move and change to conformation C_2.
3. Otherwise, *accept* the move with probability $e^{-(E(C_2)-E(C_1))/k_BT}$, where T is the absolute temperature and k_B is Boltzmann's constant.
4. If the move is not yet accepted, *reject* the move and remain in conformation C_1.

This method produces a sequence of moves with some nice statistical properties that we will cover in more depth in chapter 9. The choice of this model of dynamics once again involves a substantial oversimplification of how a chain would really fold, but it is a serviceable model for this example. This completes a model, if not a very good model, of how a protein chain will move over time.

We are now ready to formulate our initial question more rigorously. We can propose to estimate the stability of the chain as follows:

1. Place the chain into its native configuration.
2. Select the next state according to the Metropolis criterion.
3. If it is in the native configuration, record a *hit*; otherwise, record a *miss*.
4. Return to step 2.

We can run this procedure for some predetermined number of steps and use the fraction of hits as a measure of the stability of the protein. We can repeat this experiment for each mutation we wish to consider. A mutation that yields a higher percentage of hits than the original sequence over a sufficiently long simulation run is inferred to be more stable. A mutation that yields a lower percentage of hits is inferred to be less stable. This example thus demonstrates how we might use simulation to solve a biological problem.

An issue closely related to simulation is sampling: choosing a state according to some probability distribution. For example, instead of simulating a trajectory from the native state, we might repeatedly sample from the partition function defined by the energies of the states of our protein sequence. That is, we might have some probability distribution over possible configurations of the protein defined by the relative energies of the folds, then repeatedly pick random configurations from this distribution. We could then ask what fraction of states that we sample are the native state. This is actually closer to what we really want to do to solve our problem, although if we look at a lot of steps of simulation, the two approaches should converge on the same answers. In fact, simulation is often a valid way to perform sampling, although there may be much more efficient ways for some problems. For a short amino acid chain like this, for example, it might be feasible to analytically determine the probability distribution of states, given our model.

1.2.3 Parameter-Tuning

The final area of modeling and simulation we will consider is how to fit a general class of model to a specific set of data. Whether we are using a model for simulation or optimization, we will commonly have a general format for input and output, but some unknown parameters are needed to translate one to the other. We may also have a set of examples from which to learn the missing parameters. We then wish to establish the function relating inputs to outputs. A model lets us constrain the space of possible functions and judge which among the allowed ones are better explanations than others. That in turn lets us formulate a precise computational problem.

For example, suppose we want to learn about the function of a novel protease we have identified. A protease is a protein that cuts other proteins or peptides. It usually has some specificity in selecting the sites at which it cuts other proteins. That is, if it is presented with many copies of the same protein, there are some sites it will cut frequently and some it will cut rarely or not at all. Suppose we have the following examples of how the protease cleaves some known peptides:

$$SIVVAKSASK \rightarrow SASIVVAK + SASK$$

$$HEPCPDGCHSGCPCAKTC \rightarrow H + EPCPDGCH + SGCPCAKTC.$$

We can treat these examples as the input to a parameter-fitting problem. More formally, we can say our input is a set of strings on the alphabet of amino acids

$$\Sigma = \{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}$$

and a set of integer *cut sites* in each string. Our goal is to predict how this protease will act on novel sequences. Typically, we would answer this by assuming a class of models based on prior knowledge about our system, with some unspecified parameters distinguishing particular members of the class. We would then try to determine the parameters of the specific model from our class that best explain our observed data. We can then use the model with that parameter assignment to make predictions about how the protease will act on novel sequences.

We first need to define our class of models. A good way to get started is to ask what we know about proteases in general. Proteases usually recognize a small motif close to the cut site. The closer a residue is to the cut site, the more likely it is to be important to deciding where the cut occurs. A good model then may assume that the protease examines some window of residues around a potential cut site and decides whether or not to cut based on the residues in that window. The parameter-tuning problem for such a model consists of identifying the probability of cutting for any specific window. If we have a lot of training data, we may assume that the protease can consider very complicated patterns. Since our data are very sparse, though, we

probably need to assume the motif it recognizes is short and simple. That assumption is not necessarily true, and if it is not, then we will not be able to learn our model without more data. Many known proteases cut exclusively on the basis of the residue immediately N-terminal of the cut site, so for this example we will assume that the window examined consists only of that one residue.

Using these basic assumptions, we can create a formal model for cut-site prediction. As a first pass, we can assume that the probability of cutting at a given site is a function of the amino acid immediately N-terminal from that site. More formally, then, our class of models is the set of mappings from amino acids to cut probabilities,

$$f : \{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\} \rightarrow [0, 1].$$

The parameters of the model are then the 20 values $f(A), f(C), \ldots, f(Y)$ defining the function over the amino acid alphabet. This may be an acceptable model if we have sufficient data available to estimate all of these values. In this case, though, our training data are so sparse that we do not have any examples of some amino acids with which to estimate cut probabilities. So how do we predict their behavior? Once again, to answer this sort of question we have to ask what we know about our system. Specifically, what do we know about amino acids that might help us reduce the parameter space? One useful piece of information is that some amino acids are more chemically similar than others, and they can be roughly grouped into categories by chemical properties. Typical categories are hydrophobic (H), polar (P), basic (B), acidic (A), and glycine (G). If we then classify our amino acids into these groups, we end up with the following inputs:

$$PHHHHBPHPB \rightarrow PHHHHB + PHPB$$

$$BAPHPAGHBPGHHHHBPH \rightarrow B + APHPAGHB + PGHHHHBPH$$

We now have five parameters to fit in this model: $f(H)$, $f(P)$, $f(B)$, $f(A)$, and $f(G)$, that is, the probabilities of cutting after each amino acid class. In this simple model, the procedure for fitting our model to the data is straightforward: count the fraction of times a particular residue class is followed by a cut site. This procedure gives us the following parameters:

$$f(H) = 0$$

$$f(P) = 0$$

$$f(B) = 0.75$$

$$f(A) = 0$$

$$f(G) = 0$$

That answers our general question about the rules determining the behavior of this protease. In particular, we have derived what are known as *maximum likelihood estimates* of the parameters, which means these are the parameter values that maximize the probability of generating the observed outputs from our model. If we want to get more sophisticated, we can also consider how much confidence to place in our parameters based on the amount of data used to determine each one. We will also need to consider issues of validating the model, preferably on a different data set than the one we used to train it. We will neglect such issues for now, but return to them in chapter 24.

References and Further Reading

Though I am not aware of any references on the general subject matter of this chapter, the specific examples are drawn from a variety of sources in the literature. Evolutionary tree-building is a broad field, and there are many fine references to the general topic. Three excellent texts for the computationally savvy reader are Felsenstein [1], Gusfield [2], and Semple and Steel [3]. The notion of a parsimony-based tree, as we have examined it here, first appeared in the literature in a brief abstract by Edwards and Cavalli-Sforza [4]. There are many computational methods now available for inferring trees by parsimony metrics, and the three texts cited above ([1], [2], [3]) are all good references for these methods. We will see a bit more about them in chapters 2 and 3.

The use of lattice models for protein-folding applications was developed in a paper by Taketomi et al. [5], the first of a series introducing a general class of these lattice models that became known as Gō models. The specific example of a lattice move presented in figure 1.4 was introduced in a paper by Chan and Dill [6] as part of a move set called MS2. The Metropolis method, which we will cover in more detail in chapter 9, is one of the most important and widely used of all methods for sampling from complicated probability distributions. It was first proposed in an influential paper by Metropolis et al. [7].

The problem of predicting proteolytic cleavage sites is not nearly as well studied as evolutionary tree-building or protein-folding, but nonetheless has its own literature. The earliest reference to the computational problem of which I am aware is a paper by Folz and Gordon [8] introducing algorithms for predicting the cleavage of signal peptides. Much of the current interest in the problem arises from its importance in some specific medical contexts. One of these is understanding the activity of the human immunodeficiency virus (HIV) protease, a protein that is critical to the HIV life cycle and an important target of anti-HIV therapeutics. A review by Chou [9] offers a good discussion of the problem and methods in that context. Another impor-

tant application is prediction of cleavage by the proteasome, a molecular machine found in all living cells. The proteasome is used for general protein degradation, but has evolved in vertebrates to play a special role in the identification of antigens by the immune system. Its specificity has therefore become important to vaccine design, among other areas. Saxová et al. [10] conducted a survey and comparative analysis of the major prediction methods for proteasome cleavage sites, which is a good place to start learning more about that application.