2.2 DATA ANALYSIS AND PREDICTION

A major impediment to scientific progress in many fields is the inability to make sense of the huge amounts of data that have been collected via experiment or computer simulation. In the fields of statistics and machine learning there have been major efforts to develop automatic methods for finding significant and interesting patterns in complex data, and for forecasting the future from such data; in general, however, the success of such efforts has been limited, and the automatic analysis of complex data remains an open problem. Data analysis and prediction can often be formulated as search problems – for example, a search for a model explaining the data, a search for prediction rules, or a search for a particular structure or scenario well predicted by the data. In this section I describe two projects in which a genetic algorithm is used to solve such search problems – one of predicting dynamical systems, and the other of predicting the structure of proteins.

Predicting Dynamical Systems

Norman Packard (1990) has developed a form of the GA to address this problem and has applied his method to several data analysis and prediction problems. The general problem can be stated as follows: A series of observations from some process (e.g., a physical system or a formal dynamical system) take the form of a set of pairs,

$$\{(\overline{x}^1, y^1), \ldots, (\overline{x}^N, y^N)\},$$

where $\overline{x}^i = (x_1^i, \ldots x_N^i)$ are independent variables and y^i is a dependent variable (1*didN).* For example, in a weather prediction task, the independent variables might be some set of features of today's weather (e.g., average humidity, average barometric pressure, low and high temperature, whether or not it rained), and the dependent variable might be a feature of tomorrow's weather (e.g., rain). In a stock market prediction task, the independent variables might be $\overline{x} = (x(t_1), x(t_2), \ldots x(t_n)),$, representing the values of the value of a particular stock (the "state variable") at successive time steps, and the dependent variable might be $y = x(t_n + k)$, representing the value of the stock at some time in the future. (In these examples there is only one dependent variable y for each vector of independent variables $\overline{x}$; a more general form of the problem would allow a vector of dependent variables for each vector of independent variables.)

Packard used a GA to search through the space of sets of conditions on the independent variables for those sets of conditions that give good predictions for the dependent variable. For example, in the stock market prediction task, an individual in the GA population might be a set of conditions such as

$$\begin{aligned} C = \{ & (\$20 \leq \mathrm{Pr}\, ice\, of\;\; Xerox\, stock\, on\, day 1) \\ & \wedge (\$25 \leq \mathrm{Pr}\, ice\, of\;\; Xerox\, stock\, on\, day\, 2 \leq \$27) \\ & \wedge (\$22 \leq \mathrm{Pr}\, ice\, of\;\; Xerox\, stock\, on\, day\, 2 \leq \$25)\} \end{aligned}$$

where "$\wedge$" – is the logical operator "AND" This individual represents all the sets of three days in which the given conditions were met (possibly the empty set if the conditions are never met). Such a condition set C thus specifies a particular subset of the data points (here, the set of all 3-day periods). Packard's goal was to use a GA to search for condition sets that are good predictors of *something* – in other words, to search for condition sets that specify subsets of data points whose dependent-variable values are close to being uniform. In the stock market example, if the GA found

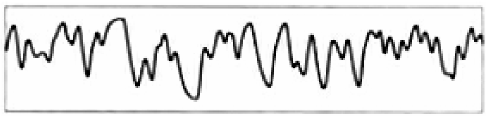

Figure 2.11: Plot of a time series from Mackey-Glass equation with $\ddot{A} = 150$. Time is plotted on the horizontal axis; $x(t)s$ is plotted on the vertical axis. (Reprinted from Martin Casdagli and Stephen Eubank, eds., Nonlinear Modeling and Forecasting; © 1992 Addison-Wesley Publishing Company, Inc. Reprinted by permission of the publisher.)

a condition set such that all the days satisfying that set were followed by days on which the price of Xerox stock rose to approximately $30, then we might be confident to predict that, if those conditions were satisfied today, Xerox stock will go up.

The fitness of each individual $\underline{C}$ is calculated by running all the data points $(\overline{x}y)$ in the training set through C and, for each x that satisfies C, collecting the corresponding *y.* After this has been done, a measurement is made of the uniformity of the resulting values of *y.* If the *y* values are all close to a particular value $\dot{A}$, then C is a candidate for a good predictor for y – that is, one can hope that a new x that satisfies C will also correspond to a *y* value close to $\dot{A}$. On the other hand, if the *y* values are very different from one another, then x satisfying C does not seem to predict anything about the corresponding y value.

As an illustration of this approach, I will describe the work done by Thomas Meyer and Norman Packard (1992) on finding "regions of predictability" in time series generated by the Mackey-Glass equation, a chaotic dynamical system created as a model for blood flow (Mackey and Glass 1977):

$$\frac{dx}{dt} = \frac{ax(1-\tau)}{1+[x(t-\tau)]^c} - bx(t)$$

Here $x(t)$ is the state variable, t is time in seconds, and a , b , c and $\ddot{A}$ are constants. A time series from this system (with $\ddot{A}$ set to 150) is plotted in figure 2.11.

To form the data set, Meyer and Packard did the following: For each data point i, the independent variables $\overline{x}^i$ are 50 consecutive values of $x(t)$ (one per second):

$$\overline{x}^i = (x_1^i, x_2^i, \ldots, x_{50}^i)$$

The dependent variable for data point i, y^i is the state variable $t^`$ time steps in the future: $y^i = x^i_{50+t`}$. Each data point $(\overline{x}^i, y^i)$ is formed by iterating the Mackey-Glass equation with a different initial condition, where an initial condition consists of values for $\{x_1 \ddot{A}, \ldots, x_0\}$.

Meyer and Packard used the following as a fitness function:

$$f(C) = -\log_2\left(\frac{\sigma}{\sigma_0}\right) - \frac{\alpha}{N_C}$$

Where σ is the standard deviation of the set of $y`s$ for data points satisfying $C$, $\sigma_0$ is the standard deviation of the distribution of $y`s$ over the entire data set, N_C is the number of data points satisfying condition C, and $\pm$ is a constant. The first term of the fitness function measures the amount of information in the distribution of $y`s$ for points satisfying C, and the second term is a penalty term for poor statistics – if the number of points satisfying C is small, then the first term is less reliable, so C should have lower fitness. The constant $\pm$ can be adjusted for each particular application.

Meyer and Packard used the following version of the GA:

1. Initialize the population with a random set of $C`s$.
2. Calculate the fitness of each C.
3. Rank the population by fitness.
4. Discard some fraction of the lower-fitness individuals and replace them by new $C`s$ obtained by applying crossover and mutation to the remaining $C`s$.
5. Go to step 2.

(Their selection method was, like that used in the cellular-automata project described above, similar to the "$\frac{1}{4}+$ >>" method of evolution strategies.) Meyer and Packard used a form of crossover known in the GA literature as "uniform crossover" (Syswerda 1989). This operator takes two Cs and exchanges approximately half the "genes" (conditions). That is, at each gene position in parent A and parent B, a random decision is made whether that gene should go into offspring A or offspring B. An example follows:

$$\textit{Parent A}: \{(3.2 \le x_6 \le 5.5) \wedge (0.2 \le x_8 \le 4.8) \wedge (3.4 \le x_9 \le 9.9)\}$$
$$\textit{Parent B}: \{(6.5 \le x_2 \le 6.8) \wedge (1.4 \le x_4 \le 4.8) \wedge (1.2 \le x_9 \le 1.7) \wedge (4.8 \le x_{16} \le 5.1)\}$$

$$\textit{Offsping A}: \{(3.2 \le x_6 \le 5.5) \wedge (1.4 \le x_4 \le 4.8) \wedge (3.4 \le x_9 \le 9.9)\}$$
$$\textit{Offsping B}: \{(6.5 \le x_2 \le 6.8) \wedge (0.2 \le x_8 \le 4.8) \wedge (1.2 \le x_9 \le 1.7) \wedge (4.8 \le x_{16} \le 5.1)\}$$

Here offspring A has two genes from parent A and one gene from parent B. Offspring B has one gene from parent A and three genes from parent B.

In addition to crossover, four different mutation operators were used:

Add a new condition:

$$\{(3.2 \le x_6 \le 5.5) \wedge (0.2 \le x_8 \le 4.8)\}$$
$$\to \{(3.2 \le x_6 \le 5.5) \wedge (0.2 \le x_8 \le 4.8) \wedge (3.4 \le x_9 \le 9.9)\}$$

Delete a condition:

$$\{(3.2 \le x_6 \le 5.5) \wedge (0.2 \le x_8 \le 4.8) \wedge (3.4 \le x_9 \le 9.9)\}$$
$$\to \{(3.2 \le x_6 \le 5.5) \wedge (0.2 \le x_8 \le 4.8)\}$$

Broaden or shrink a range:

$$\{(3.2 \le x_6 \le 5.5) \wedge (0.2 \le x_8 \le 4.8)\}$$
$$\to \{(3.9 \le x_6 \le 4.8) \wedge (0.2 \le x_8 \le 4.8)\}$$

Shift a range up or down:

$$\{(3.2 \le x_6 \le 5.5) \wedge (0.2 \le x_8 \le 4.8)\}$$
$$\rightarrow \{(3.2 \le x_6 \le 5.5) \wedge (1.2 \le x_8 \le 5.8)\}$$

The results of running the GA using these data from the $\ddot{A} = 150$ time series with $t^{`} = 150$ are illustrated in Figure 2.12 and Figure 2.13. Figure 2.12 gives the four highest-fitness condition sets found by the GA, and figure 2.13 shows the four results of those condition sets. Each of the four plots in figure 2.13 shows the trajectories corresponding to data points $(\bar{x}^i, y^i)$ that satisfied the condition set. The leftmost white region is the initial 50 time steps during which_the data were taken. The vertical lines in that region represent the various conditions on x.

$\bar{x}$ given in the condition set. For example, in plot a the leftmost vertical line represents a condition on x_{20} (this set of trajectories is plotted starting at time step 20), and the rightmost vertical line in that region represents a condition on x_{49}. The shaded region represents the period of time between time steps 50 and 200, and the rightmost vertical line marks time step 200 (the point at which the y^i observation was made). Notice that in each of these plots the values of y^i fall into a very narrow range, which means that the GA was successful in finding subsets of the data for which it is possible to make highly accurate predictions. (Other results along the same lines are reported in Meyer 1992.)

These results are very striking, but some questions immediately arise. First and most important, do the discovered conditions yield correct predictions for data points outside the training set (i.e., the set of data points used to calculate fitness), or do they merely describe chance statistical fluctuations in the data that were learned by the GA? Meyer and Packard performed a number of "out of sample" tests with data points outside the training set that satisfied the evolved condition sets and found that the results were robust – the y' values for these data points also tended to be in the narrow range (Thomas Meyer, personal communication).

Exactly how is the GA solving the problem? What are the schemas that are being processed? What is the role of crossover in finding a good solution? Uniform crossover of the type used here has very different properties than single-point crossover, and its use makes it harder to figure out what schemas are being recombined. Meyer (personal communication) found that turning crossover off and relying solely on the four mutation operators did not make a big difference in the GA's performance; as in the case of genetic programming, this raises the question of whether the GA is the best method for this task. An interesting extension of this work would be to perform control experiments comparing the performance of the GA with that of other search methods such as hill climbing.

$$C_1 = \left\{ \begin{matrix} (x_{20} > 1.122) \wedge (x_{25} < 1.330) \wedge (x_{28} > 1.168) \wedge \\ (x_{39} < 1.342) \wedge (x_{41} > 1.304) \wedge (x_{49} > 1.262) \end{matrix} \right\} \rightarrow y = 0.18 \pm 0.014$$

$$C_2 = \left\{ \begin{matrix} (x_{25} < 1.330) \wedge (x_{26} > 1.177) \wedge (x_{31} > 1.127) \wedge \\ (x_{38} < 1.156) \wedge (x_{40} < 1.256) \wedge (x_{46} > 1.194) \wedge \\ (x_{47} < 1.311) \wedge (x_{49} > 1.070) \end{matrix} \right\} \rightarrow y = 0.27 \pm 0.019$$

$$C_3 = \left\{ \begin{matrix} (x_{24} > 0.992) \wedge (x_{29} < 1.190) \wedge (x_{30} > 1.020) \wedge \\ (x_{34} < 1.090) \wedge (x_{40} < 0.951) \wedge (x_{42} > 0.599) \wedge \\ (x_{43} > 0.591) \wedge (x_{49} < 0.763) \wedge (x_{50} > 0.576) \end{matrix} \right\} \rightarrow y = 1.22 \pm 0.024$$

$$C_4 = \left\{ \begin{matrix} (x_{20} < 0.967) \wedge (x_{22} < 1.049) \wedge (x_{26} > 0.487) \wedge \\ (x_{29} < 1.066) \wedge (x_{31} > 0.416) \wedge (x_{34} < 1.008) \wedge \\ (x_{37} < 1.331) \wedge (x_{40} < 0.941) \wedge (x_{41} > 0.654) \wedge \\ (x_{42} > 0.262) \wedge (x_{48} > 0.639) \wedge (x_{49} < 0.814) \end{matrix} \right\} \rightarrow y = 1.34 \pm 0.034$$

Figure 2.12: The four highest-fitness condition sets found by the GA for the Mackey-Glass system with $\ddot{A} = 150$. (Adapted from Meyer and Packard 1992.)

To what extent are the results restricted by the fact that only certain conditions are allowed (i.e., conditions that are conjunctions of ranges on independent variables)? Packard (1990) proposed a more general form for conditions that also allows disjunctions (('s); an example might be

$$\{[(3.2 \le x_6 \le 5.5) \vee (1.1 \le x_6 \le 2.5)] \wedge [0.2 \le x_8 \le 4.8]\}$$

where we are given two nonoverlapping choices for the conditions on x_6. A further generalization proposed by Packard would be to allow disjunctions between sets of conditions.

To what extent will this method succeed on other types of prediction tasks? Packard (1990) proposes applying this method to tasks such as weather prediction, financial market prediction, speech recognition, and visual pattern recognition. Interestingly, in 1991 Packard left the Physics Department at the University of Illinois to help form a company to predict financial markets (Prediction Company, in Santa Fe, New Mexico). As I write this (mid 1995), the company has not yet gone public with their results, but stay tuned.

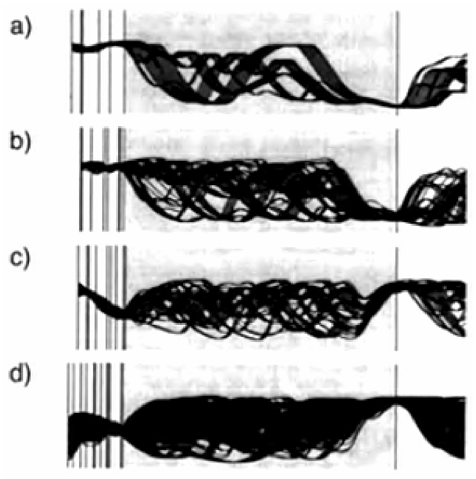

Figure 2.13: Results of the four highest-fitness condition sets found by the GA. (See figure 2.12.) Each plot shows trajectories of data points that satisfied that condition set. The leftmost white region is the initial 50 time steps during which data were taken. The vertical lines in that region represent the various conditions $\bar{x}$ on given in the condition set. The vertical line on the right-hand side represents the time at which the prediction is to be made. Note how the trajectories narrow at that region, indicating that the GA has found conditions for good predictability. (Reprinted from Martin Casdagli and Stephen Eubank (eds.), Nonlinear Modeling and Forecasting; © 1992 Addison-Wesley Publishing Company, Inc. Reprinted by permission of the publisher.)

Predicting Protein Structure

One of the most promising and rapidly growing areas of GA application is data analysis and prediction in molecular biology. GAs have been used for, among other things, interpreting nuclear magnetic resonance data to determine the structure of DNA (Lucasius and Kateman 1989), finding the correct ordering for an unordered group of DNA fragments (Parsons, Forrest, and Burks, in press), and predicting protein structure. Here I will describe one particular project in which a GA was used to predict the structure of a protein.

Proteins are the fundamental functional building blocks of all biological cells. The main purpose of DNA in a cell is to encode instructions for building up proteins out of amino acids; the proteins in turn carry out most of the structural and metabolic functions of the cell. A protein is made up of a sequence of amino acids connected by peptide bonds. The length of the sequence varies from protein to protein but is typically on the order of 100 amino acids. Owing to electrostatic and other physical forces, the sequence "folds up" to a particular three-dimensional structure. It is this three-dimensional structure that primarily determines the protein's function. The three-dimensional structure of a Crambin protein (a plant-seed protein consisting of 46 amino acids) is illustrated in figure 2.14. The three-dimensional structure of a protein is determined by the particular sequence of its amino acids, but it is not currently known precisely how a given sequence leads to a given structure. In fact, being able to predict a protein's structure from its amino acid sequence is one of the most important unsolved problems of molecular biology and biophysics. Not only would a successful prediction algorithm be a tremendous advance in the understanding of the biochemical mechanisms of proteins, but, since such an algorithm could conceivably be used to *design* proteins to carry out specific functions, it would have profound, far-reaching effects on biotechnology and the treatment of disease.

Recently there has been considerable effort toward developing methods such as GAs and neural networks for automatically predicting protein structures (see, for example, Hunter, Searls, and Shavlik 1993). The relatively simple GA prediction project of Steffen Schulze-Kremer (1992) illustrates one way in which GAs can be used on this task; it also illustrates some potential pitfalls.

Schulze-Kremer took the amino acid sequence of the Crambin protein and used a GA to search in the space of possible structures for one that would fit well with Crambin's amino acid sequence. The most straight-forward way to describe the structure of a protein is to list the three-dimensional coordinates of each amino acid, or even each atom. In principle, a GA could use such a representation, evolving vectors of coordinates to find one that resulted in a plausible structure. But, because of a number of difficulties with that representation (e.g., the usual crossover and mutation operators would be too likely to create physically impossible structures), Schulze-Kremer instead described protein structures using "torsion angles" – roughly, the angles made by the peptide bonds connecting amino acids and the angles made by bonds in an amino acid's "side chain." (See Dickerson and Geis 1969 for an overview of how three-dimensional protein structure is measured.) Schulze-Kremer used 10 torsion angles to describe each of the *N* (46 in the case of Crambin) amino acids in the sequence for a given protein. This collection of *N* sets of 10 torsion angles completely defines the three-dimensional structure of the protein. A chromosome, representing a candidate structure with *N* amino acids, thus contains *N* sets of ten real numbers. This representation is illustrated in figure 2.15.

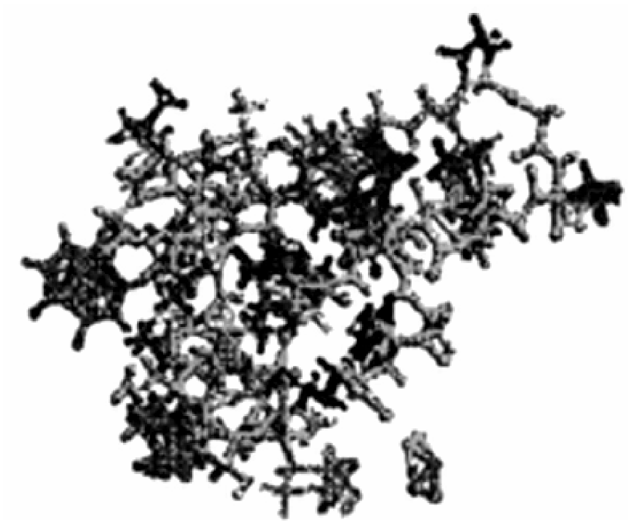

Figure 2.14: A representation of the three-dimensional structure of a Crambin protein. (From the "PDB at a Glance" page at the World Wide Web URL *http://www.nih.gov/molecular modeling/pdb at a glance*.)

The next step is to define a fitness function over the space of chromosomes. The goal is to find a structure that has low potential energy for the given sequence of amino acids. This goal is based on the assumption that a sequence of amino acids will fold to a minimal-energy state, where energy is a function of physical and chemical properties of the individual amino acids and their spatial interactions (e.g., electrostatic pair interactions between atoms in two spatially adjacent amino acids). If a complete description of the relevant forces were known and solvable, then in principle the minimum-energy structure could be calculated. However, in practice this problem is intractable, and biologists instead develop approximate models to describe the potential energy of a structure. These models are essentially intelligent guesses as to what the most relevant forces will be. Schulze-Kremer's initial experiments used a highly simplified model in which the potential energy of a structure was assumed to be a function of only the torsion angles, electrostatic pair interactions between atoms, and van der Waals pair interactions between atoms (Schulze-Kremer 1992). The goal was for the GA to find a structure (defined in terms of torsion angles) that minimized this simplified potential-energy function for the amino acid sequence of Crambin.

In Schulze-Kremer's GA, crossover was either two-point (i.e., performed at two points along the chromosome rather than at one point) or uniform (i.e., rather than taking contiguous segments from each parent to form the offspring, each "gene" is chosen from one or the other parent, with a 50% probability for each parent). Here a "gene" consisted of a group of 10 torsion angles; crossover points were chosen only at amino acid boundaries. Two mutation operators designed to work on real numbers rather than on bits were used: the first replaced a randomly chosen torsion angle with a new value randomly chosen from the 10 most frequently occurring angle values for that particular bond, and the second incremented or decremented a randomly chosen torsion angle by a small amount.

Torsion angles

amino acid 1		amino acid 2		...	amino acid 46
φ:	66.3°	φ:	-27.2°		
ψ:	45.2°	ψ:	23.1°		.
ω:	180.0°	ω:	180.0°		.
χ^1:	-22.7°	χ^1:	111.4°		.
χ^2:	127.1°	χ^2:	120.2°		
χ^3:	-100.0°	χ^3:	-22.1°		
χ^4:	32.2°	χ^4:	32.2°		
χ^5:	-125.9°	χ^5:	-87.3°		
χ^6:	55.4°	χ^6:	-95.2°		
χ^7:	76.6°	χ^7:	-54.1°		

chromosome:

[66.3 45.2 180.0 -22.7 127.1 -100.0 32.2 -125.9 55.4 76.6] ...

Figure 2.15: An illustration of the representation for protein structure used in Schulze-Kremer's experiments. Each of the N amino acids in the sequence is represented by 10 torsion angles. (See Schulze-Kremer 1992 for details of what these angles represent.) A chromosome is a list of these N sets of 10 angles. Crossover points are chosen only at amino acid boundaries.

The GA started on a randomly generated initial population often structures and ran for 1000 generations. At each generation the fitness was calculated (here, high fitness means low potential energy), the population was sorted by fitness, and a number of the highest-fitness individuals were selected to be parents for the next generation (this is, again, a form of rank selection). Offspring were created via crossover and mutation. A scheme was used in which the probabilities of the different mutation and crossover operators increased or decreased over the course of the run. In designing this scheme, Schulze-Kremer relied on his intuitions about which operators were likely to be most useful at which stages of the run.
The GA's search produced a number of structures with quite low potential energy – in fact, much

lower than that of the actual structure for Crambin! Unfortunately, however, none of the generated individuals was structurally similar to Crambin. The snag was that it was too easy for the GA to find low-energy structures under the simplified potential energy function; that is, the fitness function was not sufficiently constrained to force the GA to find the actual target structure. The fact that Schulze-Kremer's initial experiments were not very successful demonstrates how important it is to get the fitness function right – here, by getting the potential-energy model right (a difficult biophysical problem), or at least getting a good enough approximation to lead the GA in the right direction.

Schulze-Kremer's experiments are a first step in the process of "getting it right." I predict that fairly soon GAs and other machine learning methods will help biologists make real breakthroughs in protein folding and in other areas of molecular biology. I'll even venture to predict that this type of application will be much more profitable (both scientifically and financially) than using GAs to predict financial markets.