in *Model-Based Reasoning: Science, Technology, Values*,
L. Magnani and N. Nersessian (eds.),
New York: Kluwer Academic/Plenum Publishers, 2002

Modelling data: Analogies in neural networks, simulated annealing and genetic algorithms

DANIELA M. BAILER-JONES[1] and CORYN A. L. BAILER-JONES[2]
1 Department of Philosophy, University of Bonn, Germany; 2 Max-Planck-Institute for Astronomy, Heidelberg, Germany

Abstract: This paper examines several analogies employed in computational data analysis techniques: the analogy to the brain for artificial neural networks, the analogy to statistical mechanics for simulated annealing and the analogy to evolution for genetic algorithms. After exploring these analogies, we compare them to analogies in scientific models and highlight that scientific models address specific empirical phenomena, whereas data analysis models are application-neutral: they can be used whenever a set of data meets certain formal requirements, regardless of what phenomenon these data pertain to. Through the analogy, computational data analysis techniques inherit a conceptual idea from which the principle of the technique is developed. In all cases of computational data analysis techniques, the analogies used – and the metaphors generated by them – help us to understand the technique by providing a more concrete framework for understanding what is otherwise an abstract method. In the different examples, however, the significance of the analogies varies. Analogy can, though need not, be indispensable for a technique.

1. THE TOPIC

Doing science is about studying empirical phenomena and hence involves collecting data about these phenomena. Today this often involves assembling very large amounts of data, made possible through the automation of technical processes and the availability of computational support. Large amounts of data are in principle beneficial for the purpose of testing empirical claims. Large amounts of data can, however, present a major difficulty for analysis. The larger a data set gets the bigger the problem of how to interpret it meaningfully and lucidly. This is especially the case when the data are multidimensional, i.e. represented by many different variables. Once the data exist in more than three dimensions (i.e. variables), it is no longer possible simply to draw a plot and inspect it visually. On the other hand, once a pattern is found in the data and features of a phenomenon are discerned, then this finding is all the more significant when extracted from a large set of data. The increase in our ability to collect large amounts of data has a fundamental impact on the way science is done, and so considerable effort is invested in efficient methods for coping with such data.

To this end, various computational data analysis techniques have been developed over the last twenty years or so. Such techniques can be assembled into models of how to solve a certain kind of problem associated with a set of data. We call these *data analysis models*. For instance, artificial neural networks can provide a model with which to solve pattern recognition problems. Data analysis problems can arise from the study of nature, but need not (e.g. the travelling salesman problem). Examining data analysis models, we want to determine relationships to other types of models, especially *theoretical scientific models*. As in data analysis models, these primarily mathematical tools are used to deal with data that have been extracted from empirical phenomena. (Other types of scientific models, e.g. of spatial configurations, exist, but are not of immediate interest in our study.) Moreover, as in scientific models where the use of analogies is well-established, analogies are also encountered in computational data analysis techniques, which makes a comparison yet more desirable.

The point of departure for our study is our curiosity about the use of analogies in the context of computational data analysis techniques. The vocabulary used to describe these techniques is full of evocative metaphors, and some of the analogies employed have been instrumental in developing the techniques.

– *Artificial neural networks* exploit an analogy to the human brain. The idea behind artificial neural networks was to transfer the idea of parallel distributed processing, as found in the brain, to the computer in order to take advantage of the processing features of the brain.
– *Simulated annealing* is a method of optimisation, for example of determining the best fit parameters of a model based on some data. The physical process of annealing is one in which a material is heated to a high temperature and then slowly cooled. Annealing provides a framework in which to avoid local minima of energy states in order to reach the global minimum.

– *Genetic algorithms*, another optimisation technique, employ operations that mimic natural evolution to search for the fittest combination of 'genes', i.e. the optimal solution to a problem.

In Section 2, we describe these techniques in some more detail and spell out the analogies underlying them. This enables us to assess how deep these analogies go and how important they are conceptually for establishing the technique. The metaphors derived from such analogies could just be fancy talk, an attempt to advertise the method in question and to make it sound 'sexy'. Yet, we shall illustrate that there is more behind the analogies (and metaphors) typical of computational data analysis techniques.

In Section 3, we consider data analysis models in the wider context of modelling in science. We emphasize in particular that computational data analysis techniques can serve to analyse data from an enormous range of contexts or phenomena – which is why we call them *application-neutral*. This is in contrast to more conventional theoretical scientific models where the models employ theoretical concepts and physical laws that are deemed relevant for the specific physical problem in question. As we demonstrate, this fundamental difference between data analysis models and theoretical scientific models has repercussions for the role of analogies in either case.

Section 4 highlights the critical role of analogies in data analysis techniques, especially in comparison to the well-studied use of analogies in theoretical scientific models.

2. THE DATA ANALYSIS TECHNIQUES AND THEIR ANALOGIES

Data analysis techniques comprise a whole range of tools for solving data-related problems. One such problem which is relevant to the techniques we discuss is the search for the best fit of data to some model, a so-called optimisation problem. To take a simple example, we may have some data on the variation of the temperature in the Earth's atmosphere as a function of altitude and a model for this variation which has one free parameter. The process of determining the best value for this parameter from the data can be cast as an optimisation problem in which we want to minimise the error (i.e. the difference between the model's temperature predictions and the measured values) as a function of the model parameter. A schematic example of this is shown in Figure 1, where, for one parameter, the 'error landscape' only stretches in one direction.

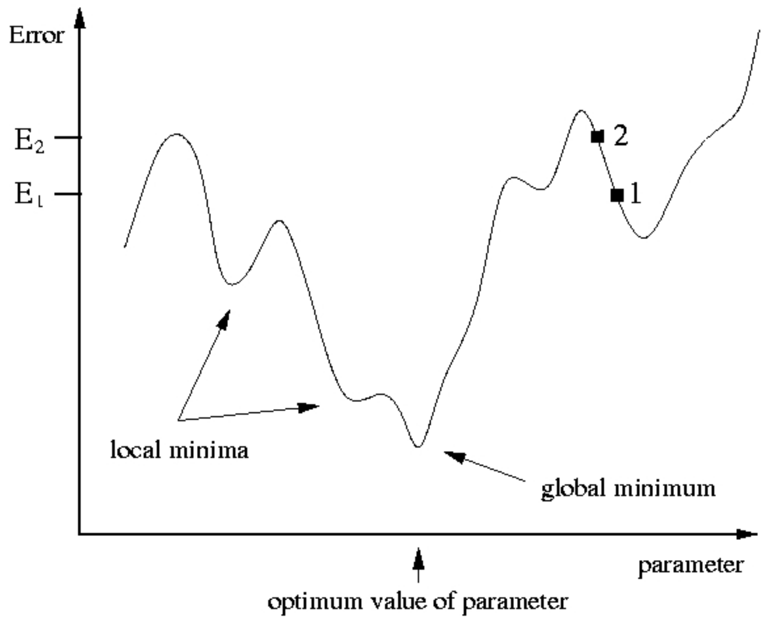

Figure 1. The error of a model's predictions as a function of its one free parameter. The error is determined from empirical data about the phenomenon that is being modelled.

The global minimum of the error is the smallest value of the error, and so corresponds to the optimum value of the parameter. In all but the simplest problems this optimum value cannot be found analytically. Thus we must use some kind of search strategy for locating it. In this single parameter example we could probably get very close to it simply by systematically evaluating the error at a large number of values of the parameter. However, many computational data analysis problems involve significantly more parameters, possibly many thousands. The optimisation problem then occurs in a much higher dimensional space and a systematic search of all parameter permutations is out of the question simply on the grounds of computer speed.

A class of search methods exist which evaluate the gradient of the error and use this in an iterative fashion to move in the 'downhill' direction of the error landscape. However, the danger of this is that the search may become stuck in local minima, which could still correspond to a large error (see Figure 1). To overcome this problem, a number of non-gradient based techniques have been developed, which frequently involve an element of random movement on the error surface. Two examples of such methods which we shall discuss are simulated annealing and genetic algorithms. Optimisation techniques such as these may, for instance, be used to train an artificial neural network.

2.1 Artificial neural networks

2.1.1 Artificial neural network models

There exist many varieties of artificial neural networks (see, for example, Hertz, Krogh, and Palmer, 1991). Here we focus on just one type, albeit one of the most widely used for data modelling. This is the supervised feed-forward neural network, which gives a functional data mapping between two data domains. To illustrate this model, consider the example problem of the classification of stellar spectra, whereby one wants to determine the underlying physical conditions (temperature, gravity and composition) of a star from its optical spectrum (Bailer-Jones, 2000). The neural network is used to give a mapping between the spectrum at its input and the physical conditions at its output (see Figure 2).

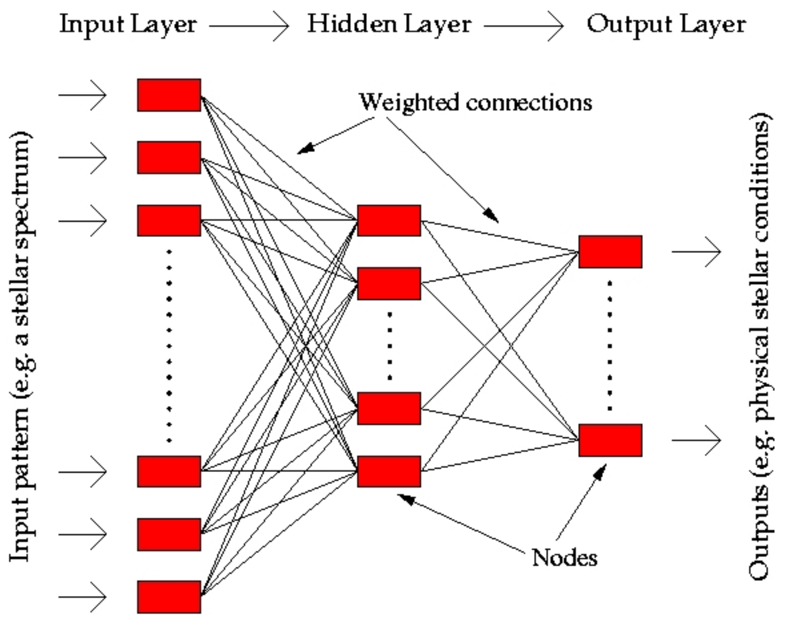
Input Layer
Hidden Layer
Output Layer
Weighted connections
Nodes
Input pattern (e.g. a stellar spectrum)
Outputs (e.g. physical stellar conditions)

Figure 2. Structure of a feed-forward artificial neural network.

Between the input and output layer, there are one or more layers of hidden nodes. These nodes combine and nonlinearly process the data passed to them from the previous layer and then pass that on to the nodes in the next layer. Each connection between the nodes has a weight associated with it which modifies the value of the data passing along that connection; these weights form the free parameters of the neural network model. For the network to give the correct mapping between the spectra and their physical parameters, these weights must be set to appropriate values.

For many high-dimensional or complex problems it is difficult to quantify the relationship accurately between the two data domains solely from physical principles, so these weights cannot be assigned according to physical principles. Instead, the weights must be inferred by 'training' the network on a set of pre-classified spectra, i.e. ones for which the ideal outputs (so-called targets) are already known. In this way the network 'learns' the relationship from the training data. Training typically proceeds by adjusting the weights until the discrepancy between the actual network outputs and the targets is minimised; this is just a (nonlinear) multidimensional optimisation problem. The network is not explicitly given any rules about which features of the spectrum are relevant for classification; it acquires this information itself from the training data. Essentially the only assumption which the network makes about the problem is that the outputs are some smooth function of the inputs.

Note that, despite any impression to the contrary that may arise from pictures such as Figure 2, the neural network is just an *algorithm*, i.e. it consists solely of equations that are used for training and applying the network. The network structure shown in Figure 2 does not really have an existence other than being a convenient way of visualising how the network operates.

2.1.2 The neural network analogy

The neural network was described above in purely mathematical terms, as a nonlinear, multidimensional parametrization model. However, neural networks are often described by analogy to the brain, hence their name. The brain excels at tasks that require the processing of a multitude of information at the same time (colours, shapes, sounds) in highly variable environments, and the brain copes with such tasks by means of many processing elements that work in parallel. The idea behind developing artificial neural networks was to transfer the idea of parallel processing to the computer in order to take advantage of some of the brain's features.

The brain consists of large numbers of neurons connected to each other by synapses. The output from the neuron is a function of its inputs from many other neurons, which are 'weighted' at the receiving synapses. This output is a nonlinear function of its input and the strength of the connection in the synapses can be modified by activity; in other words, the brain (the collection of neurons) learns (changes its synaptic weights) from experience. It is this behaviour which an artificial neural network attempts to model algorithmically, albeit in a simplified fashion. The assumption that learning occurs in the brain when modifications are made to the effective coupling between one cell and another at a synaptic junction is simulated mathematically in artificial systems through positive or negative reinforcement of connections. This forms the basis of the analogy exploited in artificial neural networks. However, it is clear that brains are considerably more complex than artificial neural networks, just on the basis that the human brain contains of order 10^{11} neurons, whereas artificial neural networks are in practice restricted to a few thousand nodes. This results in an enormous difference in the degree of complexity and computational power available. Moreover, the nodes in artificial neural networks are highly simplified in comparison to the processes at work in the neurons in the brain.

In terms of the vocabulary used in the neural network analogy, there are few terms which are carried over from the biological neural network. The terms 'nodes' and 'connections' of course have their parallels in a biological network ('neurons' and 'synapses' respectively), but it is significant that new terms have been developed for the artificial case, implying a degree of independence. The one word which is retained is 'network', although this is of course a term not restricted to the brain.

Artificial neural networks were originally inspired by attempts to provide simple models of brain function and learning (McCulloch and Pitts, 1943). Although this is still an active line of research, a significant and largely separate branch of neural computation has grown out of this which is mainly concerned with taking advantage of distributed processing for the purpose of modelling multidimensional data sets. While the analogy to the brain may have been historically relevant for the conception of such artificial neural networks, it is questionable now how much the analogy really helps to comprehend the technique in a data modelling context. Similarly, although parallel distributed processing is a central idea, often the computers on which neural networks are implemented are not parallel processors. The differences between the brain and artificial neural networks are so considerable that it is probably easier to learn directly about the specific nature of this data analysis technique, rather than to take the detour via the brain. Any analogy carries disanalogies with it. This is not a problem as long as the users of the analogy are well aware of how far the analogy stretches, i.e. regarding to which issues it can be applied with benefit. On the other hand, there is always the danger of being misled by disanalogies. In the case of the type of artificial neural network we discuss, the usefulness of the analogy is doubtful considering current uses of the technique. Thus the neural network metaphor bears the risk of misleading people into thinking that artificial neural networks are *like* the brain.

2.2 Simulated annealing

2.2.1 The technique of simulated annealing

Simulated annealing (Kirkpatrick, Gelatt Jr., and Vecchi, 1983) is a method for optimisation based on a controlled random walk on the *error surface* (the multidimensional generalisation of the error curve shown in Figure 1). Starting at some random point (1) on this surface, the error, E_1, is evaluated from the model and data. A nearby point (2) is chosen at random and the error, E_2, evaluated (see Figure 1). If the new point has a lower error, the search moves there and the process is repeated. However, if it has a higher error (as shown), there is still a chance of moving there. The probability for this is chosen to be $p = \exp(E_1 - E_2)/kT$, as the analogy to statistical mechanics suggests (Metropolis et al., 1953). This probability ranges between 1 and 0 for very small and very large error differences respectively. In other words, 'uphill' moves are permitted, albeit with decreasing probability for larger differences. This has the effect of managing to 'escape' local minima, and hence permits a more comprehensive search of the parameter space. The quantity *kT* in the equation determines exactly how probable an uphill move of a certain size is: a large *kT* makes comparatively large uphill moves more likely. The idea behind simulated annealing is to reduce *T* slowly (for a fixed *k*) as the search proceeds. This initially permits a large region to be searched. As time proceeds (and *T* is reduced), large uphill moves become increasingly prohibited, thereby focusing attention on finding what is hopefully the global minimum of the parameter space.

2.2.2 The analogy of annealing

Simulated annealing was developed in direct analogy to physical annealing, in which a material is heated to a high temperature and melted, and then slowly cooled to encourage it to solidify into its lowest energy state. The lowest energy state corresponds to a single crystal. When in the liquid form, the molecules are constantly moving around and so are able to move into different configurations, or *microstates*, which correspond to different energies of the material. If the temperature is reduced sufficiently slowly the material will visit a very large number of microstates, corresponding to thermal equilibrium. At a lower temperature, the microstates of thermal equilibrium generally correspond to lower energy states. As the temperature is reduced, the material will converge towards a microstate which is the lowest energy state.

Simulated annealing uses exactly this approach to find the lowest value of an error function – the error function is analogous to the energy in the thermodynamic case. The formula for calculating the probability of uphill moves in simulated annealing is exactly the Boltzmann formula of statistical mechanics which describes thermal equilibrium. This means that provided we visit sufficient points (microstates) on the error surface at each temperature, a situation analogous to thermal equilibrium will be maintained. Hence, as the control parameter, T (which is called 'temperature' by analogy), is slowly reduced, the lowest value of the error function will eventually be found. Of course, we would not *have* to use the Boltzmann formula to calculate the probability of uphill moves, but its choice is motivated by the analogy. If we were to take some other formula we may still be able to find the global minimum, but not in a way which could take advantage of the analogy to statistical mechanics.

The opposite of annealing is quenching, whereby a material is cooled very rapidly. In this case the material has very little opportunity to visit different configurations, so a configuration very close to that it starts in will be the one to become 'frozen' in. As the material starts off at a high temperature, it is very likely that this final configuration corresponds to a metastable, high energy state, e.g. one corresponding to a polycrystalline material. In the optimisation case, this is analogous to making very few moves before lowering the temperature. The result is that uphill moves are quickly discouraged and the search is forced to move to a local minimum in the vicinity of the starting point, without having explored the error surface.

The analogy between simulated annealing and the physics of true annealing is very close. In the former case, one tries to minimise an error as a function of a set of parameters (e.g. the weights in a neural network). In the latter case, one tries to minimise the energy as a function of the microstate or configuration, i.e. the set of particle positions. In both cases we can picture this as a multidimensional minimisation, where each dimension corresponds to a parameter in the former case and to a particle position in the latter case. The analogy arises largely because the Boltzmann formula is used in simulated annealing to calculate the probability of uphill moves. Combined with sufficient moves at each T, this ensures that a situation analogous thermal equilibrium in a physical system is achieved, and hence that the characteristics of true annealing appear in the simulated case.

Simulated annealing is a strong analogy to physical annealing, because the underlying formalism is very similar in the two cases. This is not just a useful analogy which helps us to understand the technique, but rather a situation in which the same description and equations are used. Indeed, much of our understanding of physical systems can then be applied to simulated annealing, and even concepts such as entropy and heat capacity become meaningful in the new context. There are, nonetheless, some disanalogies. In particular, a temperature has been introduced in the simulated annealing algorithm as a convenient control parameter. While it clearly has a strong analogy to physical temperature, there is no actual temperature in an optimisation context. Similarly, in all physical systems there is a universal value for the parameter, *k*, the Boltzmann constant, which is a fundamental constant of nature. In a given computational problem, on the other hand, we are free to set this, i.e. to decide what energy difference corresponds to what probability of an uphill move at a certain temperature. Nonetheless, these disanalogies are peripheral to the extent and depth of the analogy. They by no means threaten the usefulness of the analogy, because the analogy is fundamental to the technique of simulated annealing.

2.3 Genetic algorithms

2.3.1 The technique of genetic algorithms

A genetic algorithm (Holland 1975; Mitchell, 1996) is a random, yet directed search mechanism for an optimal solution to some problem. As in biological evolution, we have a population of organisms each having a different set of genes. These genes correspond to the parameters of some model we wish to optimise, so each organism represents a potential solution to the optimisation problem. For each organism we can determine how well its parameters solve the problem in hand: this determines the ‘fitness’ of each organism, with better (lower error) solutions corresponding to higher fitness. The idea behind a genetic algorithm is then to produce new members of the population from members of the current population using various *genetic operators*. The process of *recombination* (or *crossover*) is a means of ‘sexual reproduction’ in which two parents interchange genes to create two new children. The operation of *mutation* is a means of ‘asexual reproduction’ in which a child is created from a single parent by randomly changing the value of one of its genes. (There are many ways in which these operators can be implemented, but a typical example is shown in Figure 3). Thus new generations are created iteratively, and at each stage the fitness of each member is a measure of how well each solves the optimisation problem. The goal of this evolutionary approach is to produce diversity within the population and so explore various gene combinations, or solutions to the optimisation problem. However, this is not a blind search: the probability of reproduction is related to the fitness of the parents, i.e. fitter parents are generally made to produce more children. Nevertheless, not only the fittest individuals produce children: although that may seem the best strategy, it is similar to only permitting downhill moves in the search for the global minimum of the error function. As we have seen, such a strategy is prone to getting stuck in local minima. Thus, while less fit members may be produced in the interim, the goal is that, over many generations, the fitness landscape (error surface) is comprehensively explored, and that the population evolves towards a fitter state.

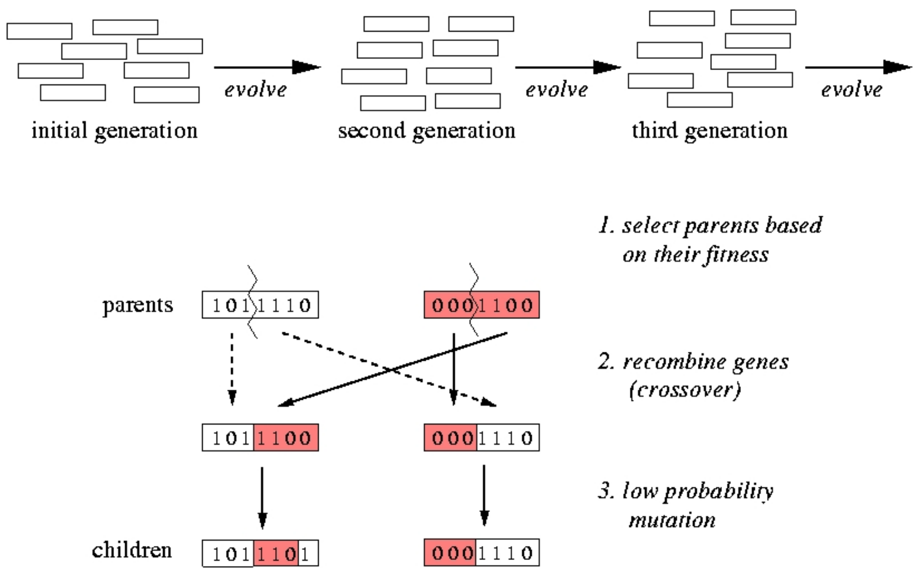

Figure 3. Demonstration of the major functions in a genetic algorithm. Starting from an initial population, two parents are selected with a probability depending on their fitness. The genes are split at some point and recombined to produce two new organisms, a process referred to as *crossover*. There is also a small chance that any one gene can mutate. The resulting children form members of the next generation, and the process is repeated until a second generation in size equal to the initial population is produced. The whole procedure is iterated for many generations with the expectation that fitter organisms are produced.

An important aspect of genetic algorithms is the encoding of the optimisation problem into a genetic formalism. In particular, the parameters of the problem need to be encoded as discrete genes, and the error function expressed as a fitness which can be determined from these genes. For example, a genetic algorithm could be used to determine the optimal weights of a neural network: each weight is represented as a gene, and the fitness is the negative of the error function.

2.3.2 The genetic analogy

The analogy with biological evolution hardly needs to be stated. It is more than evident from the vocabulary that is used in describing the method of genetic algorithms: genetics, population, organism, generation, fitness, sexual and asexual reproduction, selection, mutation. It also seems that the conception of the processes bringing about evolutionary progress are critical for the conception of this computational method: for example, the genetic operators of crossover and mutation were inspired by evolutionary processes which occur in nature. This is in contrast to the case of neural networks where the analogy to the brain may have had heuristic value, but now seems to be largely historical. The genetic algorithms technique is based on the assumption that, because in nature the analogue of these processes appear to produce an improvement in the fitness of organisms, they will do so also in a simulated environment. On the other hand, the analogy breaks down at the point of encoding. While strings of numbers may be manipulated just like genes, the way the genetic – or other – information is stored in a gene or in a string of numbers is fundamentally different. To encode information in a string of numbers involves active interference by someone who decides how best to encode some information. Another interesting disanalogy is that in biological evolution the fitness of an organism is very much dependent on the other organisms in the environment: if there are many other organisms which can eat your food (or you), then your fitness is going to be lower. This is in contrast to most optimisation problems where there is some static function which needs to be optimised, independent of the other candidate solutions.

There can be no doubt that our comprehension of the method of genetic algorithms benefits greatly from the familiarity with evolutionary concepts. This is supported by the abundance of suggestive metaphorical vocabulary. The idea for constructing genetic algorithms on the basis of the analogy to evolutionary biology requires making a considerable mental transition, in part because the encoding mechanisms are so different in the two cases. The way in which genes are manipulated, combined and expressed is very different in the biological and the genetic algorithm cases. This is unlike the case of simulated annealing where the abstract situation of the two analogues is very similar. Thus in some sense, although simulated annealing has much more in common with physical annealing than genetic algorithms do with biological evolution, simulated annealing is a much less surprising analogy precisely *because* it replicates true annealing so closely. In many ways the concepts and language carried over to the optimisation case remain the same, and are more than just an aid for dealing with the problem. With genetic algorithms, on the other hand, there is a much greater 'distance' between mathematically encoded optimisation and the field of evolutionary biology from which the inspiration for the method is derived. Consequently, the language and concepts transferred are much more subject to reinterpretation. For example, a gene and a numerical encoding called a gene are not the same. Reaping the benefit of the genetic analogy first requires reinterpretation before the surprising possibilities of the analogy can be exploited.

3. COMPARING DATA ANALYSIS MODELS WITH THEORETICAL SCIENTIFIC MODELS

Both data analysis models and theoretical scientific models are there to solve a problem, one to solve a problem of data analysis, the other to solve a problem of describing an empirical phenomenon. Of course, solving a problem about an empirical phenomenon also involves considering data, and in principle, the same set of data could provide the basis for either a data analysis model or a scientific model. First, we discuss substantial differences between the two types of models, and then explore the impact these differences have on the use of analogies in each type of model.

3.1 The contrast between theoretical scientific models and data analysis models

Both theoretical scientific models and data analysis models employ mathematical strategies for solution. A theoretical scientific model is one which has been developed employing scientific theories and concepts and which applies to a specific phenomenon. An example is a model for the temperature variation in the Earth’s atmosphere, where one wants to determine how the temperature in the atmosphere varies with distance from the Earth. This can be done largely on the basis of theoretical assumptions taken from theories of radiation transfer and thermodynamics and the properties of gases. The core of the model then comes from these assumptions, perhaps only taking into account a minimum of data, which then allows the modeller to solve certain equations. Whether the data set is large or small, one is interested in determining the values of some important, physically-meaningful parameter or parameters with the aim of better understanding the physics and chemistry of the atmosphere. Computational data analysis models similarly have the goal of modelling the relationship between two domains, such as stellar spectra and the physical conditions within stars. Again, the optimal values of some parameters in a model (e.g.

the weights in a neural network) can be determined from some data. However, the interest here is not explicitly the values of these parameters. Instead, the primary interest is in training a model to make predictions, that is, to be able to determine the physical conditions in other stars based on their spectra, using the knowledge intrinsically present in the training data. The model parameters are a means to an end and not necessarily of physical significance themselves. This is because these models are conceived without regard to the theories and concepts of the various problems to which they can be applied. The models are sufficiently general so that they can be applied to a wide class of problems with only general requirements having to be met. In consequence, beyond the goal of accurate prediction, the scientific insight that computational data models give in a specific case may be limited. A physical understanding of exactly what aspects of the physical conditions in a star give rise to which parts of the spectrum is useful, but secondary when the main goal is accurate classification, with whatever degree of model complexity that is necessary (within feasible limits).

There is a fairly straightforward reason for the fact that artificial neural networks do not provide much physical understanding. The discussed data analysis techniques are not specific to the type of data that are modelled. The techniques are designed to be independent of specific applications – they are *application-neutral*. They are *specific to a type of data analysis problem, but not to the nature of the specific empirical problem*. For example, a neural network could be applied to almost any problem of mapping between two data domains. From which empirical phenomenon the data derive, or if they even derive from an empirical phenomenon, is secondary, just as long as the problem representation conforms to some very general requirements. A theoretical scientific model is, in contrast, specific to a type of phenomenon. The theoretical concepts and laws that give shape to the theoretical model are chosen on the basis of the physical properties of the phenomenon to be modelled. These models are therefore restricted in their range of application. For example, a general relativistic model for planetary orbits is not much use in problems in which gravity is not a central factor.

Scientific models and data analysis models do not exclude each other. It is possible to model a set of data either applying physical theories or applying a neutral data analysis technique (Figure 4). For example, in metallurgy, there exists a theoretically motivated equation for the determination of the size of microscopic crystal grains in a metal which has been mechanically processed (e.g. forged). Given knowledge of the material properties, the grain sizes can be analytically calculated from the details of how the material was mechanically processed, the processing parameters. However, it is a difficulty of this approach that some of these material properties must be determined from expensive and time-consuming electron microscopy. On the other hand, a data modelling technique such as a neural network can be trained to predict the grain sizes for given processing parameters without ever needing to explicitly know these material properties, as they are intrinsically present in the required training data (Sabin, Bailer-Jones, and Withers, 2000). Measuring the grain sizes of materials processed in a certain way to construct this training data set is cheap and quick in comparison to determining the material properties, thus offering one distinct advantage of this approach.

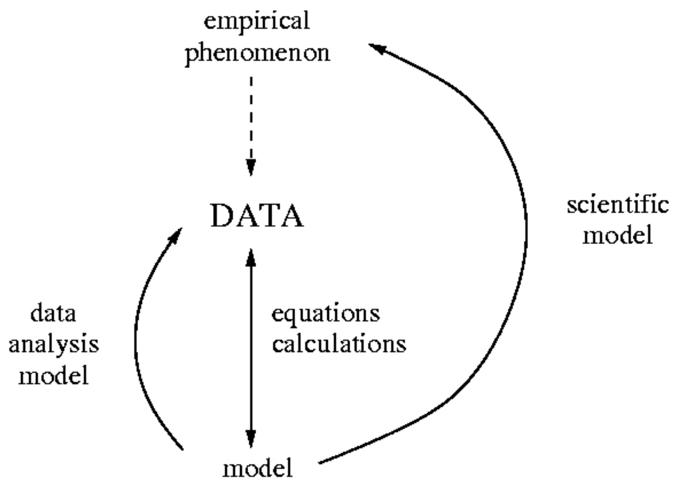

Figure 4. While data is derived from a phenomenon, the computational data analysis model refers to the data only, while the scientific models refers to the phenomenon. That data of a phenomenon are not the same as the phenomenon has been argued convincingly by Bogen and Woodward (1988).

Optimisation techniques, such as those discussed, are application-neutral not only in the sense that they can be applied to a wide range of very different problems. In turn, a problem of a certain type can be addressed by different techniques. For example, a given optimisation problem could be tackled either with simulated annealing or a genetic algorithm. Correspondingly, a computational data analysis technique can be designed in many different ways to solve a certain problem. The choice of design is not unambiguously determined by the problem, which is why an analogy to inspire the technique can be chosen fairly freely and certain specifications within it may be arbitrary.

On account of their application-neutrality, computational data analysis techniques must *encode* or parametrize the specific problem to which the technique is to be applied. This encoding needs to be done in a way suitable for the technique. For example, with a neural network we must define appropriate inputs and outputs; with an optimisation technique we must determine a suitable energy function (in the case of simulated annealing) or fitness function (in the case of genetic algorithms). For a theoretical scientific model, this encoding is intrinsic to the very conception of the model because the model itself is motivated by the physics of the problem, so no separate encoding procedure is required to make the model match the problem. Only for an application-neutral model does the information about a *specific* application need to be encoded in the model. What is more, only the necessary minimum of information is encoded.

It is a consequence of the generality of many computational data analysis techniques that they involve an element of arbitrariness. This is exemplified by the control parameters (e.g. the number of hidden nodes in a network, the value of k in simulated annealing, or the rate of mutation in genetic algorithms) which generally have to be decided on by the user. Different control parameters can be tested for efficiency, speed and reliability, but there is often little in the problem application which suggests the appropriate values. Even the free parameters of the model which are determined from the data (e.g. the weights in a neural network) retain some degree of arbitrariness, because different parameter settings may give equally good model predictions.

3.2 Analogies in data analysis models

Theoretical scientific models have a relationship to the phenomenon modelled that is more direct than in the case of application-neutral data analysis models. Consequently, if an analogy is used in a theoretical model, then this is an analogy between phenomena. For instance, in the 1840s, William Thomson, the later Lord Kelvin, modelled electrostatic attraction in terms of thermal conduction. He ‘recycled’ the mathematical tools that had already been developed for conduction for the exploration of the more recent subject of electrostatics. This led to viewing thermal conduction and electrostatic attraction in analogy to each other (Harman, 1982, p. 29), a perspective which subsequently triggered further important insights into electrostatic phenomena. The mathematical analogy which Thomson exploited highlighted a *physical analogy* between the phenomena of thermal conduction and of electrostatic attraction. That analogies are important for the development of scientific models is a long-standing theme in philosophy of science (e.g. Hesse, 1966; Achinstein, 1968; for an overview see Bailer-Jones, 2001). When analogies feature in scientific modelling, then one empirical phenomenon is viewed in terms of another phenomenon, i.e. *in analogy to* that latter phenomenon. The expectation is that the interpretation of one empirical phenomenon may benefit from the interpretation of another (perhaps comparable) empirical phenomenon. If so, a transfer of insights gained from the study of the latter to the former becomes possible.

In contrast to this, the analogy that is employed in artificial neural networks – the analogy to the human brain – has nothing to do with the specific phenomenon to which that technique is applied, i.e. the stellar spectra that may be classified with the help of an artificial neural network have nothing to do with and are not analogous to the human brain. If artificial neural networks exploit the analogy to the human brain, then this by no means implies a connection to the phenomenon that is modelled by the network: A physical analogy between the analogue used in the model (e.g. the brain) and the phenomenon modelled (e.g. the formation of stellar spectra) is neither required not likely because of application-neutrality. Equally, if the optimisation technique used in the network is simulated annealing, then annealing also has no link to stellar spectra, nor, for that matter, to the way the brain works. Each of these analogies is separate in that it only illustrates or inspires a *technique*, and not a phenomenon that may be modelled with the help of that technique.

When, for the development of a theoretical scientific model (e.g. of electrostatic attraction), an analogy (e.g. to thermal conduction) is exploited, then the analogy is the source and motivation for the choice of the equations. Something similar can be said for artificial neural networks. The analogy to the brain is, at least historically, responsible for the kind of mathematical algorithm that the network employs. But the impact of an analogy on the equations is not always equally strong. Artificial neural networks only carry a vague inheritance from the true complexity of neurons, whereas in our example of simulated annealing, the impact of the analogy on the equations and concepts used in the algorithm is very significant. There, equations are transferred directly and they are employed in essentially the same way as in statistical mechanics. Thus, the impact of an analogy on the development of a computational data analysis technique can vary from example to example.

In sum, in the case of data analysis techniques, the analogies employed illuminate the technique to a greater or lesser degree, but do not illuminate the specific problem to which the technique is intended to provide a solution. However, precisely because the technique is developed independently of applications, it is important that the technique can be illustrated independently of these. Having a separate and application-independent analogy at one's disposal is likely to facilitate access to the technique for a whole range of different users with different backgrounds. Talk about annealing, temperature etc. provides common ground when applying the technique in different areas. Different users of artificial neural networks, genetic algorithms and simulated annealing may share an understanding of the analogies to the brain, to annealing and to evolutionary change, but possibly nothing more.

4. SO, WHY ANALOGIES?

Just as analogies are prevalent in theoretical scientific models, they also have an important – but different – role to play in computational data analysis models. There are a number of uses for a concrete analogy backing up an abstract data analysis tool. The first and foremost is to provide an idea for developing the tool, an idea for an algorithm appropriate for attacking a certain kind of problem (parallel distributed processing in the case of artificial neural networks, evolution and genetic changes in the case of genetic algorithms, the Boltzmann distribution in the case of simulated annealing).

Second, there arises the need to communicate about these abstract tools. There is a metaphorical language associated with analogies employed in models that helps to put into words what otherwise seems abstract and inaccessible for want of familiarity. Martin and Harré (1982) call this a 'spin off' of the analogy when metaphors matching the analogy are introduced, but this expression may betray the conceptual importance of the vocabulary generated (Bailer-Jones, 2001): temperature in simulated annealing or fitness in genetic algorithms. Because the computational data analysis techniques themselves are application-neutral, metaphors based on analogies can be seen as constantly maintaining the link to the analogy in use. The analogy, with the metaphors attached to it, may then help to cope with the abstractness of the technique. Because the analogy only illuminates the technique and not a phenomenon to which the technique is applied, it provides an aid in overcoming the difficulty of abstractness without involving any notions about the specific application. The metaphors generated by analogies underlying computational data analysis techniques further a sense of familiarity and of concreteness in connection with the otherwise abstract computational tools. They help to attach concrete ideas to tools that are intrinsically without concrete application.

Third, there is likely to be a teaching element in the use of analogies and metaphors. Because computational data analysis techniques may be applied in a wide range of different areas, users may come to them from quite different disciplinary backgrounds. Those who enter into the area of computational data analysis may do so simply because, in their own discipline, they have a problem to solve that happens to lend itself to that type of analysis. Consequently, there will be a significant group of people who require a basic understanding of these methods, or need relatively easy access to the central ideas in order to understand better the algorithms used. Such learners depend heavily on conceptual aids linking the abstract to something more concrete.

In short, while analogies in computational data analysis techniques vary in their depth and in the extent to which they generate descriptive metaphors, analogies have an impact well beyond the original conception of computational data analysis techniques. Just like analogies in scientific models, they live on to serve not only the experts during conception and development of the technique, but perhaps as much the more casual user during acquisition and use.

5. REFERENCES

Achinstein, P., 1968, *Concepts of Science*, John Hopkins Press, Baltimore, Maryland.

Bailer-Jones, C.A.L., 2000, Stellar parameters from very low resolution spectra and medium band filters. T_{eff}, log g and [M/H] using neural networks, *Astronomy & Astrophysics* 357:197-205.

Bailer-Jones, D. M., 2001 (to appear), Models, Metaphors and Analogies, in: *Blackwell Guide to Philosophy of Science*, P. Machamer and M. Silberstein, eds., Blackwell, Oxford.

Bogen, J., and Woodward, J., 1988, Saving the phenomena, *The Philosophical Review* 97:303-352.

Harman, P. M., 1982, *Energy, Force and Matter*, Cambridge University Press, Cambridge.

Hertz, J., Krogh, A., and Palmer, R.G., 1991, *Introduction to the Theory of Neural Computation*, Addison-Wesley, Wokingham.

Hesse, M., 1966, *Models and Analogies in Science*, University of Notre Dame Press, Notre Dame.

Holland, J.H., 1975, *Adaptation in Artificial and Natural Systems*, University of Michigan Press, Ann Arbor.

Kirkpatrick, S., Gelatt Jr., C.D., and Vecchi, M.P., 1983, Optimization by simulated annealing, *Science* 220:671-680.

McCulloch, W.S., and Pitts, W.S., 1943, A logical calculus of ideas immanent in nervous activity, *Bulletin of Mathematical Biophysics*, 5:115-133.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., and Teller, E., 1953, Equation of state calculations for fast computing machines, *Journal of Chemical Physics* 21:1087-1092.

Mitchell, M., 1996, *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, Massachusetts.

Martin, J., and Harré, R., 1982, Metaphor in science, in: *Metaphor*, Miall, D. S., ed., The Harvester Press, Sussex, pp. 89-105.

Sabin, T.J., Bailer-Jones, C.A.L., and Withers, P.J., 2000, Accelerated learning using Gaussian process models to predict static recrystallization in an Al-Mg alloy, *Modelling and Simulation in Materials Science and Engineering* 8:687-706.