Journal of Intelligent & Fuzzy Systems 16 (2005) 233–238
IOS Press

Book Reviews

Design of Logic-Based Intelligent Systems, Klaus Truemper, Wiley, 2004, ISBN 0-471-48403-2

Many people have heard about NP-hard problems. Usually, when somebody mentions that a problem is NP-hard, it means that this problem is, in general, very difficult to solve. In other words, usually, NP-hardness is bad news.

However, more and more often, researchers are discovering that NP-hardness may be good news as well. To understand how NP-hardness can be good news we need to go into some detail on what exactly is NP-hardness.

The notion of NP-hardness is based on several basic notions.

The first such notion is the notion of a *feasible algorithm*. Everyone understand what an algorithm is – it is a well-defined sequence of computational steps, described in such a (boring) detail that a computer can easily follow them without any human supervision. Computer scientists have developed many algorithms. Some of them – like many sorting algorithms or efficient algorithms for Fourier transform – are used all the time. Other algorithms are only described in textbooks, where they are shown on simple examples, but they are rarely – if ever – used in practice. Why? Because for reasonable size data, these algorithms require a practically impossible number of computational steps.

Since the early days of computers, computer scientists realized that some algorithms are practical (*feasible*) and some are not feasible, but this difference has always been very difficult to describe in precise terms. As of now, the best formalization is the notion of a *polynomial-time* algorithm: if the running time of an algorithm can be bounded by a polynomial of the length n of the input, the algorithm is feasible; otherwise, it is not. For example, an algorithm that requires quadratic time n^2 is feasible, cubic-time algorithms are feasible, but an exponential time algorithm, that requires 2^n steps, is not: even for a reasonable size $n \approx 200$, we need more computational steps than there are particles in the entire Universe.

This description is not perfect: e.g., an algorithm that requires time $10^{100} \cdot n$ is clearly not practical, but, according to the above definition, it is feasible. However, for most algorithms, polynomial-time indeed means practical and vice versa.

The second notion is the notion of a problem from the *class NP*. In many real-life problems, it may be difficult to find a solution, but once we have a candidate for a solution, it is easy to check whether this candidate is a solution or not. For example, when we solve a complex system of non-linear equations, it may be difficult to find a solution, but once we have values, we can easily check whether these values constitute a solution: it is sufficient to plug these values into the equations and check the equalities. We have already mentioned that feasible means polynomial time. As a result, such problems are called "non-deterministic polynomial", where "non-deterministic" means that we can make guesses. Non-deterministic polynomial (NP, for short) means that once we have a candidate for a solution, we can check whether this candidate is indeed a solution in polynomial time.

The third notion is the notion of *reduction*. Reducing a general Problem A to another general Problem B means that we have an efficient way to match each particular case a of the Problem A to some case(s) b of the Problem B in such a way that from the solution to b, we can efficiently produce the solution to a. Example: multiplication can be reduced to computing squares – if we can compute the square of a number fast, then we can compute $x \cdot y$ as

$$\frac{(x+y)^2 - (x-y)^2}{4}.$$

If Problem A can be reduced to Problem B, this means that Problem B is, in general, at least as hard as the problem A.

It turns out that there are some problems to which all other NP problems can be reduced. Such problems are at least as hard as any problem from the class NP, so they are called *NP-hard*.

Historically the first example of an NP-hard problem is the problem of *propositional satisfiability* (SAT, for

short). The input to this problem is a propositional formula, i.e., anything that can be obtained from "yes"-"no" (propositional) variables $x_1, \ldots, x_n$ by using logical connectives "and" (&), "or" ($\vee$), and "not" ($\neg$). Example: $(x_1 \vee x_2)\,\&\,(x_1 \vee \neg x_2)$. Given such a formula, we would like to either find values of the variables that make it true, or return the message that no such values are possible.

Here comes the good news. Since SAT is NP-hard, an arbitrary Problem A from the class NP can be reduced to it. Thus, if we have a good package that efficiently solves many instances of SAT, then, through this reduction, we may be able to efficiently solve many instances of the Problem A.

In addition to problems from the class NP, in many practical applications, we also encounter *optimization problems*. In an optimization problem, we must find the alternative with the largest value of a given objective function. These problems are not in the class NP because once we have a candidate solution, it is not clear how we can check its optimality without comparing it with all possible alternatives.

For optimization problems, we can also establish the notion of reduction, and prove that an appropriate optimization version of SAT (MINSAT) is the hardest. Thus, if we have a good algorithm for solving many instances of MINSAT, we can therefore solve many instrances of other optimization problems.

The author describes many cases when the problems faced by a logic-based intelligent system can be naturally reduced to SAT and MINSAT, and where, therefore, the existing efficient SAT and MINSAT packages help in solving the intelligent system problems. Example of such successful applications include computer-aided design, automatic music composition, management of hazardous materials, traffic control, credit rating, voice recognition, and medical diagnostics.

The description is very clear. The reader should be warned, however, of two things.

First, the exposition is very textbook-like, motivating examples do not appear until Chapter 11. A reader should be warned not to give up before that.

Second, the title is somewhat misleading. Logic-based intelligent systems is half of AI, the author does not talk about them at all, the book is only about the systems in which problems are reduced to SAT and MINSAT. Even among thus reduced systems, the author only talks about his own research, leaving aside a vast amount of literature about, e.g., smodels, a tool that efficiently uses SAT-solving algorithms for deduction in Prolog-type AI knowledge bases.

As long as the readers understands these two things, the book reads well and can be highly recommended.

Vladik Kreinovich
Book Review Editor
Journal of Intelligent & Fuzzy Systems

Nonlinear Dynamic Modeling of Physiological Systems, by Vasilis Z. Marmarelis, IEEE Press and J. Wiley, Hoboken, New Jersey, 2004, ISBN 0-471-46960-2

Many physiological processes are highly non-linear. There has been a lot of research in which nonlinear dynamic models are used to describe such processes, e.g., in neural networks, but usually, such research uses parametric models. The corresponding models are typically very approximate; they may adequately describe the important qualitative features of, say, neurons or of visual perception but because of their approximate nature, they often lead to a poor quantitative accordance with the empirical data – and quantitative predictions are important. For example, while it is important to understand the qualitative mechanisms behind diabetes-related changes in insulin and blood sugar levels, it is even more important to be able to predict how exactly these levels will change in a specific patient – and thus, to apply necessary control. In view of this importance, the author advocates the use of *non-parametric* methods of nonlinear dynamics, methods in which, instead of restricting ourselves to specific models with a small finite number of parameters, we consider a potentially infinite number of parameters that can describe, in principle, an arbitrary nonlinear dynamic system.

This well-written book provides a detailed introduction into non-parametric techniques for handling nonlinear dynamical systems. The main intended audience of this book is people who are interested in physiological applications, so it provides a nice introduction into signal processing for physiologists, and a nice history of physiology for scientists and engineers. The level of the understandability is so great that I would heartily recommend this book to anyone who is interested in nonlinear dynamics, no matter where their potential application interests are.

The main idea behind non-parametric techniques in nonlinear dynamic systems is similar, e.g., to the use of polynomials in static nonlinear models: if the linear dependence between the input x and the output y is not adequate, then we can consider quadratic models $y \approx a_0 + a_1 \cdot x + a_2 \cdot x^2$, cubic models, etc. It is known that an arbitrary continuous function on a bounded domain can be approximated, within any given accuracy, by a polynomial of appropriate order. Similarly, an arbitrary continuous dependence of the output signal $y(t)$ on the inputs signal $x(t)$ can be, in general, described, e.g., by a dynamic analogue of polynomials – Volterra series $y(t) = a_0(t) + \int a_1(t,s) \cdot x(s)\, ds + \int a_2(t,s,s') \cdot x(s) \cdot x(s')\, ds\, ds' + \ldots$

From the purely mathematical viewpoint, Volterra models (or equivalent Wiener models based on the Fourier domain approach) are universal approximators. In principle, based on the observations, we can use the Least Squares techniques (or the corresponding robust techniques if the distribution is not Gaussian), determine the parameters of these models, and then use the resulting model to make forecasts. This has been efficiently done for nonlinear systems in science and engineering, but for physiology applications, there is a need to modify these techniques:

First, it is desirable to have parameters whose meaning would be clearer to the physiologists. For example, Fourier-type descriptions are natural in engineering where a sinusoid wave input is normal, but in physiology, a typical input is a rather a short-term signal (a "wavelet"). It is therefore desirable to use coefficient w.r.t. some physiologically reasonable bases – e.g., with respect to Laguerre polynomials.

Second, it is desirable to use simple algorithms that can be implemented on easily accessible low-level PCs (that, say, doctors can carry into the field) rather than on stationary high-performance computers that are normally used in the analysis of nonlinear dynamic systems like ocean or atmosphere. To handle this problem, the author shows how the ideas of efficient training algorithms from artificial neural networks can be extended to training general connectionist models of general nonlinear dynamic systems (which, as the author shows, are equivalent, e.g., to the general Volterra models).

A special emphasis is placed on non-Gaussian error distributions. In engineering and science, most frequently, the measurement errors are Gaussian. The mathematical reason for this empirical fact is the Central Limit Theorem, according to which, crudely speaking, the distribution of the sum of many small independent error components is close to Gaussian. So, when we eliminate the major sources of error, we get closer and closer to the Gaussian distribution. In physiology, we have to deal with serious measurement errors: e.g., EEG is affected by the patient's skull. It is, in principle, possible to insert a sensor inside the skull and get a better measurement result with a nice error distribution, but it is desirable to extract as much information from the non-invasive lower-quality measurements before subjecting the patient to additional invasive medical procedures. The author suggests the use of exponential Weibull-type distribution with the probability density proportional to $\exp(-|e|^d)$; for such distributions, the Maximum Likelihood Method leads to

$\sum |e_i|^d \to \min$ – a natural generalization of the Least Squares Method to $d \neq 2$.

All this theoretical background is described in Chapters 1 through 4. Chapter 5 provides a step-by-step practitioner guide to using the models. Chapter 6 describes examples of physiological applications to various phenomena in the neurosensory, cardiovascular, renal, and metabolic-endocrine systems. For all these phenomena, the match between the model's predictions and the actual observations is very impressive.

Chapter 7 describes how these techniques should be modified to handle the cases of several inputs, several outputs, and the case of spatiotemporal modeling that is important, e.g., in modeling visual perception. Chapter 8 deals with biological neurons, while Chapter 9 handles non-stationary systems.

Overall, this book is perfect as a research tool, as a reference book, and even as a textbook. I highly recommend it to everyone who is interested in nonlinear dynamics.

Vladik Kreinovich
Book Review Editor
Journal of Intelligent & Fuzzy Systems

Computationally Intelligent Hybrid Systems: The Fusion of Soft Computing and Hard Computing, Seppo J. Ovaska (Ed.), IEEE Press and J. Wiley, Hoboken, New Jersey, 2005, 0-471-47668-4

In many engineering applications, we know the exact characteristics of the control system, and we can use the traditional ("hard computing") techniques to produce an optimal control. In many other applications, the only information that we have about the system is examples of inputs and corresponding outputs; in this case, we can use neural networks to learn the system and to control it properly.

In yet other cases, we have the experience of experts who successfully control such systems (e.g., fly a helicopter); the experts can often only formulate their control rules by using words from the natural language such as "if the robot is close to an obstacle, it should slow down". To describe this knowledge and use it in automatic control, we can use the methodology of fuzzy control, the methodology that was specifically designed to handle such rules. Neural networks and fuzzy logic are examples of "soft computing", when we use heuristic intelligent methods instead of traditional optimization-based algorithms.

Both traditional hard computing methods and the soft computing methods have many useful applications. However, these methods correspond to the two extreme cases: when we have a complete information about the controlled system and when we have very little information about it. In many practical cases, we have a partial knowledge of the situation. For example, in addition to the expert rules, we could have a partial model of the controlled system. In principle, we can still ignore the partial model and base our control on the expert rules only, but it would be beneficial to use the partial knowledge to improve the quality of the resulting control. In other words, what we need is a fusion of soft computing and hard computing.

For example, in some cases, we know a class of system to which a given real-life system belongs, but we do not know the exact values of the parameters corresponding to this particular system. In this case, we can use the general parametric formula for the optimal control of such system, and then use the neural network techniques to determine the values of the parameters based on the observed behavior of the system.

The book, edited by one of the active promoters of the fusion between hard and soft computing, provides an overview of different types of such fusion, and describes many interesting practical applications of such a fusion to flight control, to control of electric motors, to tool wear monitoring, to power systems, to computer security, and to data mining.

The book is very well written, it is quite accessible for practical engineers and at the same time quite interesting for theoretical computer scientists. I hope that it will inspire more people to use the (currently under-utilized) fusion techniques.

Vladik Kreinovich
Book Review Editor
Journal of Intelligent & Fuzzy Systems

Smart Environments: Technology, Protocols, and Applications, Diane J. Cook and Sajas K. Das (Eds.), Wiley, Hoboken, New Jersey, 2005, ISBN, 0-471-544 48-5

Computers are now so efficient and small that it is possible to embed computers into our every day environment, both at the office and at home, to make these embedded computers communicate with each other and make intelligent decisions – in other words, to turn our offices and homes into smart environments. For example, at home, we may want all the electronic devices from temperature control to entertainment systems to home security to be intelligent, connected, and self-repairing.

These smart environments are already appearing, but there is still a lot of interesting challenges – how to set up sensors, how to provide communications, how to provide security, etc. This edited book describes the state-of-the-art in different aspects of smart environments. It provides an encyclopedic coverage of many possible aspects.

Of course, with such a wide coverage, there is no way to go into detail on any of these topics. As a result, the chapters of this book are mainly general surveys, with lots of references to technical papers but mostly with no explicit formula, algorithms, or technical details. For example, Chapter 2 – about wireless sensor networks – mentions wavelets, but does not have any explanation of what wavelets are or how to handle them. There are a few formulas in Chapters 9 and 12, but these formulas are mainly to illustrate the point, not to explain what exactly is done and how.

Overall, the book provides a nice general overview of different aspects of smart environments, so that a reader can get a general impression of the state of the field. It also has technical references for those who want to know more about specific aspects.

Vladik Kreinovich
Book Review Editor
Journal of Intelligent & Fuzzy Systems