The Book Review Column[1]

by William Gasarch
Department of Computer Science
University of Maryland at College Park
College Park, MD, 20742
email: gasarch@cs.umd.edu

In this column we review the following books.

1. **Handbook of Graph Theory** Edited by Gross and Yellen. Reviewed by William Gasarch. This is a massive (over 1000 pages) handbook that alleges to cover most of modern Graph Theory. Does it? Does it cover the parts relevant to theoretical computer scienct? If the reviewer were to ask random people to supply him with random questions about graph theory, and the reviewer looked them up, how well would the book do? Read the review and find out.

2. **Reasoning about Uncertainty** by Joseph Y. Halpern. Review by Wenzhong Zhao. This book examines formal systems for representing uncertainty and advocates the use of *plausibility measures* for representing uncertainty. The author considers the updating of beliefs based on the changing information, especially when the new information contradicts the agent's old belief.

3. **Learning Kernel Classifiers: Theory and Algorithms** by Ralf Herbrich, and **Learning with Kernels: Support Vector Machines, Regularization Optimization and Beyond** by Bernhard Schölkopf and Alexander J. Smola. These books are reviewed jointly by Luc T. Wille The two books under review cover kernel classification and the closely related support vector machines (SVMs). The key idea is to perform classification through the construction of an inner product function, called a kernel, which allows one to quickly map any input pattern to one element of a set of target patterns, without the explosion in computational time that would normally occur if the number of training patterns increases.

4. **Essentials of Constraint Programming** by T. Frühwirth and S. Abdennadher Reviewed by Carlos Oliveira. Constraint programming (CP) is an area of computer science in which the considered problems can be expressed as a set of logical rules, also known as constraints. This results in a huge class of problems, encompassing domains such as logic programming, mathematical programming, and optimization. The book aims to be self-contained and easy to read.

[1] © William Gasarch, 2004.

Books I want Reviewed

If you want a FREE copy of one of these books in exchange for a review, then email me at gasarchcs.umd.edu

Reviews need to be in LaTeX, LaTeX2e, or Plaintext.

Books that Defy Classificatoin

1. Turing (A novel about computation) by Christos Papadimitriou. (This is really a novel!)

Books on Algorithms

1. *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica* by Pemmaraju and Skiena.
2. *Algorithms: Design Techniques and Analysis* by Alsuwaiyel.
3. *Computational Techniques of the Simplex Method* by Maros.
4. *Immunocomputing: Principles and Appliations* by Tarakanov, Skormin, Sokolova.
5. *Computational Line Geometry* by Pottmann and Wallner.
6. *Data Structures, Near Neighbor Searches, and Methodoolgy: Fifth and Sixth DIMACS Implementation Challenges* Edited by Michael Goldwasser, David Johnson, Catherine McGeoch.

Books on Cryptography

1. *Data Privacy and Security* by David Salomon.
2. *Elliptic Curves: Number Theory and Cryptography* by Larry Washington.
3. *Block Error-Correcting Codes: A Computational Primer* by Xambo-Descamps.

Misc Books

1. *Tolerance Graphs* by Golumbic and Trenk.
2. *Dynamic Reconfiguration: Architectures and Algorithms* by Vaidyanathan and Trahan.
3. *Logic for Learning* by Lloyd.
4. *Combinatorial Designs: Constructions and Analysis* by Stinson.

Review of *Handbook of Graph Theory*[2]
Edited by Gross and Yellen
Publisher: CRC, 2004
1192 pages, $119.95, Hardcover

Review by William Gasarch

1 Introduction

One uses a handbook by looking up things that you always wanted to know or that come up. Hence, I decided to review this handbook by asking a random collection of theorists (the editors of SIGACT NEWS and theorists in the Maryland-Wash DC area) to email me questions they are either curious about or think ought to be in a handbook. I comment on how well the book does on answering each one, and then summarize how well the book did overall. For each question asked I looked rather carefully in the table of contents and the index; hence, if I say 'the book did not have anything on topic X' and in reality it does, then the books organization is at fault.

There are 11 sections, each one of which is broken into chapters. There are 54 chapters total. For example, one of the sections is "Algebraic Graph Theory" and one of its chapters is "Cayley Graphs". To see the table of contents go to either www.crcpress.com or www.amazon.com. or www.graphtheory.com.

Section 2 of this review comments on the questions that the handbook did well on, Section 3 comments on those that the handbook did not do so well on, Section 4 gives a general opinion, and Section 5 talks about the broader issue of putting handbooks online (this handbook is not online). Realize that this is a subjective judgement. The book does not claim to be relevant for computer scientists. Hence if they omit (say) expander graphs for sorting this does not mean that its a bad Handbook of Graph Theory. Graph theory is a vast topic and they should be commended for even trying to write a handbook on it.

The book has very few proofs. This is fine– it has many pointers to the literature. The intent is to tell you basic defintions and facts, and where to look for more.

2 Questions the handbook did well on

1. What is known about Graph Isomorphism in terms of complexity? There is no separate chapter on this; however, it is the index and not hard to find. The book has the following information.

   (a) If the degree of both graph is bounded then GI is in P.

   (b) If the genus of both graph is bounded then GI is in P.

   (c) If the eigenvalue multiplicity of the adjacency matrix is bounded then GI is in P.

   (d) If PH does not collapse then co-GI has sub-exp proofs. (This is a recent result of Klivans and Melkebeek.)

   The book *does not* have "if GI is NPC then PH collapses" This is a very bad omission.

[2] ©2004, William Gasarch

2. What are the current values of the Ramsey Numbers? There is a chapter on Ramsey Theory. The book has an up-to-date table of Ramsey Numbers and a pointer to the dynamic survey on small Ramsey numbers, in the Electronic Journal of Combinatorics.

3. If a graph has genus g then what is its chromatic number? There is a chapter on Graph Coloring and it has the result: If a graph G has Euler Char ϵ then $\chi(G) \leq \left\lfloor \frac{7+\sqrt{49-24\epsilon}}{2} \right\rfloor$.

4. What is the fractional chromatic number of a graph? It was easy to find this out. One definition is as follows: Take the Integer Programming formulation of graph coloring. Relax it to Linear Programming. The answer is the fractional chromatic number. There are other definitions that are more in the spirit of coloring.

5. Is there much in the book about coloring infinite graphs? There is a short list of facts known about this. I'll list two here: (1) an infinite graph is k-colorable iff every finite subgraph is k-colorable, (2) If $\chi(G) = \infty$, then for every infinite arithmetic progression $A \subseteq N$, G containts a cylce whose length belongs to A. They should have included a reference for material on infinite graph colorings For example, *Infinite Combinatorics* by A. Hajnal, which is in *Handbook of Combinatorics* edited by Graham, Grotschel, Lovasz. Having said this it should also be noted that most graph theorists study finite graphs so this could be considered a topic that is okay to omit given that the authors had to make some hard choices.

6. Is there much on Edge colorings? Vizing's theorem, which states that the edge-chromatic number is either the d or $d+1$ if d is the degree, is there. It was easy to find.

7. Is there anything on Spectral Theory? Yes. There is a good chapter on the Spectral theory of graphs. This theory relates a graph to its matrices eigenvalues. We give two examples of theorems stated. (1) Let r be the largest eigenvalue. G is bipartite iff $-r$ is also an eigenvalue. (2) If H is an induced subgraph of G then largest eigenvalue of H is $\leq$ largest eigenvalue of G. Most other theorems are more about the eigenvalues then the graphs, e.g., if G is r-regular then r is an eigenvalue with eigevector $(1, \ldots, 1)$; and all of the other eigenvectors have coordinates that sum to 0.

8. What is the Girth of a graph? The book had this and it was easy to find. The girth of a graph is its shortest cycle. The concept is used in the book alot.

9. What is the strong perfect graph theorem? This is in the book and easy to find. A *hole* in a graph is a circuit of length ≥ 5 with no chords. An *anti-hole* is the complimentary graph of a hole. A graph is *perfect* if every vertex-induced subgraph H has $\omega(H) = \chi(H)$ and $\alpha(H) = \rho(H)$. ($\omega(H)$ is the size of the largest cliqu set, $\chi(H)$ is the chromatic number, $\alpha(H)$ is the size of the largest independent set, and $\rho(H)$ is size of smallest clique cover.) The *Strong Perfect Graph Theorem* states that a graph is perfect iff it has no hole or antihole.

10. Does it describe a fast algorithm for solving a weighted bipartite matching problem? Yes, it has a nice chapter on matching that includes pointers to algorithms and their runtimes.

11. Does it have material on Cycles and Pancycles? The book has the definition of pancycle, which I will call a win.

12. What is "tree width," and what are its motivations? There is a chapter on *Algorithms on Recursively Constructed Graphs* which has a subchapter of 4 pages on *Algorithms on Treewidth-k graphs*. Roughly speaking, the treewidth of a graph measures how well the graph

can be decomposed. The book mentions that for there are many (they claim 'hundreds') of linear-time algorithms for bounded treewidth graphs.

13. Do they have material on flow algorithms? Do they ever! They have a Section on *Networks and Flows* which has four chapters.

14. Is their much on planarity? The index only has two references to planarity, but the book has more material on it. There are 17 facts listed about them in one place, and throughout the book comments are made for the planar case. Their is a section on Topological Graph Theory which has material related to planarity. The theorem that Planarity is in P is not in the book.

15. Are their algorithms relevent to graphics? There is one chapter on graph drawing which, while mostly theoretical, has some material on practical approaches. Also, the chapter on Topological graph theory will be useful for practioners since this material has been found to be useful.

3 Questions the Handbook did Badly on

1. Is there anything on games on graphs? The notion of the Grundy Number of vertices in a graph is in the book, but not its connection to games. I wonder why its there at all if not for games.

2. Is there anything on expander graphs? The book had NOTHING on this. I find that amazing.

3. Is there anything on constructive lower bounds for Ramsey Numbers? No.

4. Is there anything on Infinite Ramsey Numbers? No. What is especially appaling is that there is a book on this topic (*Combintorial Set Theory: Partition Relations for Cardinals. Studies in Logica nd the Foundations of Mathematics* by Hajnal, Mate, and Rado) that could be referenced, and also *Infinite Combinatorics* by A. Hajnal, which is in *Handbook of Combinatorics* edited by Graham, Grotschel, Lovasz.

5. Is there anything on random walks? No.

6. I am interested in the topic of "packing disjoint spanning trees". Does the book have results on this problem? No.

7. What is the relationship between the expansion factor of a graph and the eigenvalues of the adjacency matrix. There was nothing on this since there was nothing on expanding in general.

8. What is a Gomory-Hu tree? The book did not have this, so I still don't know.

9. Does is talk about edge splitting? Classical edge splitting invloves removing edges (s, u) and (s, v) and adding the edge (u, v) while maintaining some nice connectivity properties. The book has material on splitting, and on connectiviey, but never the two do meet.

10. For which values of n is K_n the disjoint union of Hamiltonian cycles. They have a whole chapter on Hamiltonian graphs, but don't have this information.

11. An outerplanar graph must have at least two degree-2 nodes. They have then definition of outerplanar, but not this theorem.

12. Is there coverage of shortest-path algorithms? The book has material on dynamic all-pairs-shortest-path problems. The problem is to maintain a Data structure S so that, for all pairs, the shortest distance between them can be answered easily. The key complexities are query-time and update-time. While this is a fine problem to discuss it is far more obscure than the usual shortest-path problem which seems to not be in the book at all.

13. What is the fixed linear crossing number of graph and what is the computational complexity of finding it? There is nothing on linear crossing number (not even the definition). There is nothing on crossing number either.

14. Is there any material on Hypergraphs? They define them and give their bandwidth, but thats it.

15. What is known about separators? The answer my friend, is not in this book, the answer is not in this book.

4 Opinion

The books coverage is arbitrary. I draw your attention to too very odd contrasts.

1. They have "if PH does not collapse then co-GI has sub-exp proofs" but they do not have "if GI is NPC then PH collapses".

2. They have material on the Dynamic all-pairs-shortest-path probem but not on the standard one.

They also do not have expander graphs, which I thought was of interest to pure graph theorists as well as computer scientists.

One day there will hopefully be a *Handbook of Graph Theory for Computer Scientists*. Until that day, this book should be in the library of any school that has a computer scientist that uses graph theory. That probably covers all readers of this column.

The *Handbook of Combinatorics* is not as good for computer scientists as this book is; though they both cover much of interest.

5 A Broader Question

Consider the following facts.

1. This book is over 1000 pages so its hard to carry around.

2. The index is not very good (this is true of all handbooks I have reviewed). The reason for this may be that the index is already 25 pages long. Even so, the objection still stands.

3. Some chapters may get outdated soon.

What if this book was online? What if it was online in such a way that it was easily searchable, and easily updated? Then the above points would not longer be true. This point of view applies best fast moving fields, and hence in Computer Science more than in math. The most common objection is that the authors would not get their monetary reward since less copies would sell. This is nonsense in that the authors get very little reward anyway and are not doing it for the money.

Review of
Reasoning about Uncertainty[3]
2003
Authors: Joseph Y. Halpern
Publisher: The MIT Press

Reviewer: Wenzhong Zhao
Department of Computer Science
University of Kentucky
wzhao0@cs.uky.edu

1 Overview

Dealing with uncertain and imprecise information has been one of the major issues in almost all intelligent systems. Uncertainty is a fundamental and unavoidable feature of our daily life. It arises because agents almost never have access to the whole truth about their environment. It can also occur when the agents have incomplete and/or incorrect information about the properties of their environment.

Agents must reason and act in an uncertain world. In order to deal with uncertainty intelligently, they need to be able to represent uncertainty in an appropriate format which allows them to reason about it. There are many different ways of representing uncertainty in the literature. Among these are *probability measures*, *sets of probability measures*, *Dempster-Shafer belief functions*, *possibility measures*, *ranking functions* and *plausibility measures*. Probability is the most common and broadly-used representation of uncertainty. However, it is not necessarily the best one for all situations. The author points out the shortcomings of probability measures. Different representations of uncertainty are good for different situations. Plausibility measures can be viewed as a generalization of all others.

There are various logics of reasoning about uncertainty, either propositional or first-order or both. Different approaches to formalizing real problems, including nonmonotonic reasoning, belief change, counterfactual reasoning, problems of statistical inference, and etc., have been proposed in the literature.

There are very few books with topics specific to reasoning about uncertainty on the market. *"Reasoning about Uncertainty"*, the book under review, is a rare kind of textbook that examines formal systems for representing uncertainty and advocates the use of *plausibility measures* for representing uncertainty. The author considers the updating of beliefs based on the changing information, especially when the new information contradicts the agent's old belief. He also discusses qualitative, quantitative and plausibilistic Bayesian networks. Regarding reasoning under uncertainty, the book discusses knowledge and belief; default reasoning and the semantics of default; reasoning about counterfactuals; belief revision; first-order modal logic; and statistics and beliefs.

The author tries to make the book accessible to readers with different backgrounds, as stated in the *Preface*. Wherever possible, this book contains enough detail to make it self-contained. As any textbook should, it contains plenty of exercises, ranging from relatively simple ones, which test the understanding of key concepts studied in the book, to rather elaborate and sophisticated ones, such as proofs of some important theorems. In addition, at the end of each chapter there is a section of notes, which provide references to the materials in the chapter as well as some details on materials not covered in the chapter.

[3]©2004 Wenzhong Zhao

2 Summary of Contents

The book consists of twelve chapters, and also features an extensive bibliography and glossary of symbols at the end.

In **Chapter 1**, *Introduction and Overview*, the author tries to convince readers that reasoning about uncertainty can be subtle and that it requires a careful analysis by providing the description of several puzzles and problems, such as the *Monty Hall puzzle* [4] and the *two-coin problem* [5]. The author also addresses that in many cases there is no quantitative information available, only qualitative information.

The second part of this chapter gives an overview of the book, and provides some suggestions on how to use it as a text for semesters with different lengths. It also describes the dependencies between different chapters.

Chapter 2, *Representing Uncertainty*, describes different measures for representing uncertainty. Probability is, perhaps, the best-known and widely-used approach to representing uncertainty in a fine-grained manner. The author gives a brief review of probability to make the chapter completely self-contained. However, he notes that there are several serious problems in representing uncertainty with probability. Among these are the following three: (1) probability is not good at representing ignorance; (2) the agent may not be able to assign probabilities to all sets; and (3) the agent may not have the resources required to compute the probabilities.

Representing uncertainty with *sets of probability measures*, *Dempster-Shafer belief functions*, *possibility measures*, and *ranking functions* is discussed. All these representations are numeric. The author also introduces a general representation of uncertainty *plausibility measures*, which can be considered as a generalization of all the other representations. This representation is nonnumeric. Furthermore, the author concludes that general results regarding uncertainty can be formulated more elegantly when using plausibility measures.

Chapter 3, *Updating Beliefs*, describes how to update beliefs based on new information. However, due to different representation of uncertainty, updating beliefs depends on the way uncertainty is represented.

The author starts discussions on knowledge updating with a very simple setting, in which an agent's uncertainty is captured only by a set of possible worlds with no complex structure. The author argues that even in this simple setting, three implicit assumptions are being made: (1) the agent does not forget; (2) what the agent is told is true; and (3) the way that the agent obtains new information does not itself gives the agent information.

Conditioning is the standard way of incorporating new information in probability theory. The author uses the *second-ace puzzle* [6] as an example to illustrate how to incorporate new information

[4] The *Monty Hall puzzle* assumes the following scenario. Suppose you are on a game show and are given a choice of three doors. There is a valuable prize behind one door, while there are prizes of little value behind the others. After you made a choice, the game host would open one of the other two doors. It is always a prize of little value. You are then given an option of staying with the initial choice or switching to the other door. The question is: should you stay with your original guess, or should you switch to the other door, or does it matter?

[5] The *two-coin problem* assumes the following scenario. Suppose Alice has two coins. One of them is fair, and the other is biased with the probability of landing heads twice as high as that of landing tails. Alice chooses one of her coins (assume she can tell the difference between the two coins) and is about to toss it. Bob does not know which coin Alice has chosen or with what the probability the fair coin is chosen. The question is: what is the probability, according to Bob, that the outcome will be heads? and what is the probability according to Alice?

[6] The *second-ace puzzle* assumes the following scenario. Suppose there is a deck of four cards: the ace and deuce of hearts, and the ace and deuce of spades. After two cards are dealt to Alice, Alice told Bob that she has an ace, and then tells him that she has the ace of spades. The question is: what should the probability be, according to Bob, that Alice has both aces?

and the subtleties involved with conditioning. This chapter examines conditioning in the context of probability: *probabilistic conditioning*. One of the most important results in probability theory *Bayes's Rule* is also discussed.

As the author says, things get even more complicated when uncertainty is represented by means other than probability. The author considers analogies of conditioning for other representations discussed in the previous chapter. He even considers generalizations of conditioning when the standard conditioning does not apply. For example, *Jeffrey's Rule* gives a unique probability measure as long as the observation is consistent with the initial probability. At the end of the chapter, the relative entropy and maximum entropy are discussed.

Chapters 4, *Independence and Bayesian Networks*, introduces the notion of *independence* which is closely related to updating. It is discussed in detail in the context of probability. A more general form of independence, *conditional independence*, is also covered.

In order to get a well-structured and compact representation of uncertainty, people tend to adopt the notion of *Bayesian network*, an important tool for describing likelihood measures. Independence in Bayesian networks is discussed. The author argues that Bayesian networks can be used with other representations of uncertainty as well since the definition of Bayesian networks does not depend on the use of probability.

Chapter 5, *Expectation*, first introduces the notion of *expectation* for probability measures, then extends that notion to the other representations of uncertainty. *Decision theory*, where expectation is most relevant, is introduced as well. This chapter ends with the notion of *conditional expectation*, which can be viewed as a generalization of conditional probability.

Up to **Chapter 6**, *Multi-Agent Systems*, the author implicitly assumes that there is only one agent at any particular time, and that situations being modeled are static. This chapter deals with more dynamic aspects of belief and probability. It discusses interactions between different agents, with each reasoning about others' uncertainty. Two types of frames, the *epistemic frames* and the *probability frames* are described before the author formally introduces the *multi-agent systems* framework, which allows one to model time and multiple agents in a natural way.

After he discusses various issues regarding the representations of uncertainty in the previous chapters, in **Chapter 7**, *Logics for Reasoning about Uncertainty*, the author starts to consider formal logics for reasoning about uncertainty. These logics provide powerful tools for representing arguments where uncertainty exists, and various methods for characterizing uncertainty. This chapter begins with a brief review of propositional logic, and then introduces a number of different propositional logics for reasoning about uncertainty, such as logics for *reasoning about knowledge* (the *modal epistemic logic*), *reasoning about more quantitative notions of uncertainty - probability, quantitative likelihood and relative likelihood*, *reasoning about independence*, and *reasoning about expectation*. The choice of which propositional logic to use depends largely on the following: (1) what underlying representation of uncertainty (i.e. a probability measure or a ranking function) is used; (2) how significant quantitative reasoning is; and (3) the notions being reasoned about (i.e. likelihood or expectation).

Chapter 8, *Beliefs, Defaults, and Counterfactuals*, deals with *default reasoning* and *counterfactual reasoning*. The author introduces the notions of *belief*, *default* and *counterfactual*. The difference between defaults and counterfactuals can be captured by making different assumptions about the properties of the belief. Default reasoning involves leaping to conclusions and may be *nonmonotonic*. Counterfactual reasoning involves reaching conclusions with assumptions that may be counter to what actually occurred. The author argues that plausibility measure plays a key role in this analysis. Some of the representations of uncertainty discussed in previous chapters provide a good framework for capturing both default reasoning and couterfactual reasoning.

Chapter 9, *Belief Revision*, considers the problem of *belief revision* in a qualitative setting. The author uses the *Circuit-Diagnosis Problem* as a test bed for illustrating the issues involved in belief revision. A fundamental question is raised: how an agent should revise his/her beliefs when new information is observed, especially when the new information contradicts his/her old beliefs? Then it introduces a class of interpreted plausibility systems called *belief-change systems*. *Belief revision*, a particular type of belief change often used in the axiomatic approach, is discussed in detail. Based on this framework, belief revision can be viewed as *conditioning* as long as beliefs are represented with appropriate plausibility measures. At the end of the chapter, different types of belief revisions such as *iterated revision* and *Markovian belief revision* are introduced.

The logics considered up to **Chapter 10**, *First-order Modal Logic*, just extend the propositional logic with *modal operators* such as knowledge, belief, and probability. These logics have their limitations on expressive power inherited in any propositional logic. For example, propositional logic cannot deal with individuals, the properties they have, the relationships between them, and etc. However, these can be done in *first-order logic*. This chapter considers *first-order modal logic* which allows for both modal reasoning and first-order reasoning. This combination adds a great deal of expressive power to the language and leads to new subtleties, which makes it important to distinguish the two kinds of "probabilities" — statistical information and degrees of belief.

As discussed in the previous chapter, **Chapter 11**, *From Statistics to Beliefs*, attempts to establish a connection between statistical knowledge and degrees of belief. In this chapter, the author focuses on probabilistic reasoning, and claims that most of the ideas described here should be applicable to other representations of uncertainty. It introduces a property called *reference classes*, which is considered to be desirable in this context. After that the author describes the *random-worlds approach* and properties that all random worlds satisfy, followed by an application of this approach: default reasoning. Random-worlds approach has many attractive features, but it also has some serious problems such as *representation dependence* and *learning*.

Chapter 12, *Final Words*, completes the book by offering a summary of the key points that are discussed in the book, along with a brief discussion on each of them.

3 Opinion

This book covers much of the author's research on reasoning about uncertainty, and includes a collection of research papers that the author published during the last two decades. It examines formal ways of representing uncertainty and considers various logics for reasoning about uncertainty.

In our opinion the book achieves its goal of being a unified introduction to a certain philosophy for representing and reasoning about uncertainty, although some readers might be uncomfortable with its mathematical formalizations of ideas in terms of definitions and theorems. The book succeeds in presenting a uniform framework for describing various representations of uncertainty, updating of beliefs, and logical systems for reasoning about uncertainty. The author strongly promotes the use of plausibility measures for reasoning about uncertainty. General results regarding uncertainty can often be formulated rather elegantly in terms of plausibility measures, as stated by the author.

This book is recommended both as a textbook for senior/graduate course in computer science, artificial intelligence and economics (particularly game theory), and as a reading or reference for graduate students or researchers in areas related to mathematics, philosophy and statistics. The author includes enough detail to make it almost completely self-contained, and accessible to readers with different backgrounds. However, some previous training in both probability and propositional logic would definitely be helpful.

Review[7]

Learning Kernel Classifiers:
Theory and Algorithms
Book authored by Ralf Herbrich
Publisher: MIT Press, Cambridge, Mass., 2002
ISBN 026208306X, Price $42.00, hardcover, 384 pages

and

Learning with Kernels:
Support Vector Machines, Regularization Optimization and Beyond
Book authored by Bernhard Schölkopf and Alexander J. Smola
Publisher: MIT Press, Cambridge, Mass., 2002
ISBN 0262194759, Price $65.00, hardcover, 644 pages

Reviewed by: Luc T. Wille, Dept of Phy., Florida Atlantic Univ, Boca Raton, FL 33431

1 Introduction

Pattern recognition is arguably the critical first step in intelligence – be it natural or artificial. Science could not exist if humans were not able to spot regularities. Only subsequently do we analyze and classify them, and ultimately come up with underlying descriptions, some of which eventually make it to the lofty status of natural laws. Of course evolution did NOT ENDOw humans (or animals for that matter) with the ability to recognize patterns so we could build grandiose scientific edifices, rather we need that ability at the most elementary level. Humans simply cannot function in everyday life without pattern recognition capabilities and when those capabilities are impaired, as in patients with Alzheimer's disease for example, the result is nothing less than tragic.

What human beings generally do unthinkingly and easily has turned out to be phenomenally difficult to emulate on a computer. However, difficult does not mean impossible and, stimulated both by the sheer intellectual challenge and the potential pay-offs, researchers have tackled the pattern recognition problem with gusto. As a result, recent years have seen remarkable breakthroughs in the vast field of machine learning. Moreover, the increasing glut of information generated by society has given further impetus to the development of powerful data mining software.

The two books under review cover one of the most important tools in modern machine learning, namely kernel classification and the closely related support vector machines (SVMs). This subject matter is extensive and rapidly evolving with most of the findings buried in journals and conference proceedings, which are not always the most accessible sources for those new to the field. In addition, kernel classification is fairly complex and borrows heavily from statistics and applied mathematics. For all these reasons there is an evident need for an expository source with a strong pedagogical bend. Such a book would be welcome not only to the artificial intelligence community, but to numerous researchers in other fields, notably bioinformatics, management, physics, and statistics. Now MIT Press has given us not one, but two books with essentially the same aim.

2 Other Books

Let us first look at what else is out there. The classic source on machine learning is Mitchell's book[1] which by necessity is rather superficial in any individual area but invaluable in its coverage

[7]©2004 Luc T. Wille

of the field as a whole. However, this book does contain some material relevant to kernel based learning. Another classic, with Knuth-like cult status amongst aficionados, is Duda and Hart[2] whose new edition[3] contains a nice brief description of the subject. It is probably the best source for those who merely want to know what kernel classifiers are and how they fit in with other pattern recognition techniques. Those wishing to learn more could always go to the originator of the method, Vladimir Vapnik, whose books (for example, Ref. [4]) tend to be thought-provoking, profound, and intellectually challenging. They are great reading for those who already know the subject, but not for the faint of heart, nor for newcomers to the field. Two more accessible introductions for novices are Cristianini and Shawe-Taylor[5] and Hastie et al.[6] The former was widely lauded as particularly user-friendly, especially to graduate students, while the latter has been praised for its mathematical rigor and was highly recommended to statisticians. Interested readers might also want to check the Web site `http://www.kernel-machines.org` with which several of these authors, including the ones whose books are under review, are associated.

So how do these two new books stack up against the competition? Let us start with the thinner book (which alphabetically also comes first).

3 Learning Kernel Classifiers: Theory and Algorithms, by Herbrich

Herbrich's book starts with an introductory chapter which explains clearly and with illustrative examples, from handwriting recognition, what learning is, how it relates to inference, and what kernel classifiers are. The key idea is to perform classification through the construction of an inner product function, called a kernel, which allows one to quickly map any input pattern to one element of a set of target patterns, without the explosion in computational time that would normally occur if the number of training patterns increases. This approach can be implemented very elegantly in a Support Vector Machine which is an ingenious algorithm that keeps the training error fixed, while minimizing the confidence interval.

The remainder of the book is subdivided in two parts: one on learning algorithms and one on learning theory. Each of these merits about 100 pages. The book then concludes with 140 pages of appendices. This may seem like a bizarre arrangement, but it works! In order to enhance the readability and speed up the exposition, a great deal of background, definitions, and proofs, as well as pseudocode is relegated to these appendices. The result is a book that moves along nicely and reads well, but without sacrificing rigor. Such a treatment is particularly important in this case, because kernel classifiers rely heavily on the notation of linear algebra and Hilbert spaces, which tends to make for heavy reading. A word of warning here is that both books assume a reasonable familiarity with probability theory (with some knowledge of measure theory), a thorough understanding of linear algebra, and a smattering of functional analysis.

The first part of the book contains two chapters entitled: Kernel Classifiers from a Machine Learning Perspective, and Kernel Classifiers from a Bayesian Perspective. I found a side-by-side reading of these two chapters to be very enlightening, since the same problem is approached from two radically different angles which makes for a more profound understanding. The second part also comes in two chapters: Mathematical Models of Learning, and Bounds for Specific Algorithms. The latter chapter should also be of interest to complexity theorists. These four chapters all end with nice sections containing bibliographic remarks, which allow the author to put the material covered in a broader context and to point to further developments. The book concludes with a very useful list of symbols, an extensive bibliography, and an index. (With regard to the bibliography I

note a number of references to German translations of books that are readily available in English versions and to rather obscure references on general topics for which standard texts exist.)

The author argues that the two parts can be read independently, but this seems a bit optimistic. Readers jumping in with part 2 will find themselves referring back to part 1 quite frequently. The book originated in the author's PhD. thesis and although there has clearly been a great deal of editing, at times it still reads more like a dissertation than a genuine text book. If there is one minor peeve I have it is about the lack of applications and examples. It would have been pedagogically stronger to work through some illustrative examples to demonstrate the various methods. There are some results on standard data sets (thyroid and sonar series, for example) but these are rather isolated incidences. As it is, anybody wishing to teach a course based on this book would have to come up with their own examples. For instructors it would also have been nice if the book had included problem sets. As it is, the book is not well suited for classroom purposes. However, it is strong in mathematical precision and clarity of exposition.

4 Learning with Kernels: Support Vector Machines, etc. by Schölkopf and Smola

The other offering is the volume by Schölkopf and Smola, which is a good deal heftier than the Herbrich book, at over 600 pages and with a larger page format. The title and lay-out of the book give it more of a definitive feel and this is confirmed by the table of contents. The book also contains a number of features that should be particularly attractive to teachers. Each chapter starts with a a brief overview and a list of prerequisites, which makes for easy reading and helps with preparation through possibly review of previous chapters. Also, most chapters in the book end with a set of problems; these are marked on a three-level scale according to level of difficulty and also contain some open problems. However, quite a few of the problems are of the "complete the proof" variety, which is of rather limited value to students trying to understand the material. However, a good teacher should have no problem coming up with other assignments.

This book too starts with a tutorial introduction, although the authors present classification problems as dots in landscapes, rather than something more concrete such as an image that needs to be analyzed. There are three main parts to the book. The first one is called "Concepts and Tools" and contains five chapters: Kernels, Risk and Loss Functions, Regularization, Elements of Statistical Learning Theory, and Optimization. I found this part to be particularly well written and of value to anyone who wishes to learn about machine learning in general. Part II addresses "Support Vector Machines". It is split up in six chapters: Pattern Recognition, Single-Class Problems: Quantile Estimation and Novelty Detection, Regression Estimation, Implementation, Incorporating Invariances, and Learning Theory Revisited. The first three of these are mathematically rather heavy although the authors do a good job of guiding the reader along. The chapter on implementation is excellent and should help anyone to make the extra step from theory to computer code. The third part is devoted to "Kernel Methods", with six chapters on Designing Kernels, Kernel Feature Extraction, Kernel Fisher Discriminant, Bayesian Kernel Methods, Regularized Principal Manifolds, and Pre-Images and Reduced Set Methods. Here too, some demands are made on the reader's mathematical background, but the rewards are well worth some perseverance. Finally, the book contains a brief section with some data sets and proofs, as well as a brief overview of mathematical prerequisites. The latter is meant more as a refresher to those who already possess the prerequisites – others will have to consult one of the references given and may have to do some prior studying. An extensive list of references, an index, and a list of symbols conclude the book.

5 Comparing the Two Books

Both books are meticulously edited by MIT Press, on high-quality paper, in an excellent binding, with clear figures, and a thorough index. (The latter is a particular hang-up of this reviewer who has seen too many books loose in value because of a poor or, horror of horrors, absent index.) I spotted no mistakes and relatively few typos in either book, again testimony to the careful editing. Herbrich's Web page has an up-to-date list of errata, while the corresponding link for the Schölkopf-Smola book is, rather regrettably, empty. (At least I take it that we are not to assume that the book is error-free!)

The two books have much in common. In fact, a great deal of Herbrich is a subset of Schölkopf and Smola, but that does not make the Herbrich book superfluous. It is to be recommended for its breezy style, mathematical rigor, and focused presentation. It should appeal to theorists, but is less suited for programmers or for classroom use. Within its self-imposed boundaries it is very thorough. The Schölkopf and Smola book is more encyclopedic, insofar that is possible in a field that is very much in a state of flux. It is well suited for classroom use and contains many pointers for computer implementation. Because of its wider scope it can be criticized for areas that are not covered or only superficially so. To compensate for this the authors list subjects that were omitted and provide pointers to the relevant literature. Still some areas are missing entirely, for example work on fuzzy logic approaches (see the book by Kecman[7] not quoted in either book). Neither book is particularly well written. In both, the English is adequate and perfectly understandable, but it lacks somewhat in fluency and excitement.

6 Opinion

So, finally, what is one to recommend someone who wishes to learn about kernel classification and support vector machines? Let me offer a flow chart description. If the goal is merely to gain some understanding, look at the new Duda et al. book[3] perhaps supplemented by a visit to the Web page mentioned above. Those who wish to learn more and particularly those wishing to implement SVMs have three options. If they feel their mathematical background is rather shaky, they should peruse Cristianini and Shawe-Taylor[5], and from there move to either book under review. Those with a stronger mathematical training can directly go to these books, while readers with particular interest in statistics should check out the Hastie book.[6]

Both books are clearly aimed at researchers in artificial intelligence, more particularly those interested in machine learning and data mining. Complexity theorists will find some of the work on algorithmic bounds of interest. Other researchers who are faced with classification problems, notably in bioinformatics and image processing will be interested in these works, as will statistical physicists studying perceptrons and neural networks.

Summarizing, both books contain a wealth of material not readily accessible elsewhere in book form, presented in a fairly pedagogical manner. As the field is still rapidly evolving, readers should consult the technical literature for the latest insights, but both books can be warmly recommended as providing a firm foundation.

References

[1] T. M. Mitchell, *Machine Learning* McGraw-Hill, New York, 1997.

[2] R. O. Duda, and P. E. Hart, *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.

[3] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, (2nd Edition), Wiley, New York, 2000.

[4] V. N. Vapnik, *Statistical Learning Theory*, Wiley, New York, 1998.

[5] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, Cambridge University Press, Cambridge, 2000.

[6] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning*, Springer Verlag, Berlin, 2001.

[7] V. Kecman, *Learning and Soft Computing: Support Vector Machines, Neural Networks, and Fuzzy Logic Models*, MIT Press, Cambridge, 2001.

Book Review: Essentials of Constraint Programming [8]

Authors: T. Frühwirth and S. Abdennadher

151 page,s $35.00, Hardcover, Springer-Verlag

Review by Carlos Oliveira, Dept. of Ind. and Systems Eng., University of Florida

1 Introduction

Constraint programming (CP) is an area of computer science in which the considered problems can be expressed as a set of logical rules, also known as constraints. This results in a huge class of problems, encompassing domains such as logic programming, mathematical programming, and optimization.

In this book, the authors give a introductory view of the techniques and technologies underlying constraint programming. The book is divided into four parts discussing different aspects of the subject. The first part gives a summary of the basic concepts on which constraint programming is based. The second part of the book describes some of the most important constraint systems. The third part is concerned with the applications of constraint programming. Finally, the last part is an appendix with fundamental topics from logic.

The book has the main objective of giving a self-contained and easy to read introduction to this large area. So, it would not be fair to judge the book on the basis of completeness and depth. Most of the theorems are given without proofs, and some important results in the area of constraint programming are just mentioned.

2 Contents

In the **introduction**, the concept of constraint programming is explained, and a brief historical sequence is given, showing the main advances that lead to the development of the area. The strengths of constraint programming are emphasized, such as the fact that some classes of constraint

[8] ©2004 Carlos Oliverira

programs can be solved automatically using logic techniques. Constraint programming has also a simple and powerful language, that considerably simplifies the process of modeling problems. Thus, the main objective of the area is to find efficient techniques for solving problems posed in this elegant language, or at least a subset of it. The introduction ends with an overview of the book, describing the for divisions mentioned above.

The fist chapter of Part 1 (**Algorithm = Logic + Control**) is a couple of paragraphs describing the constraint programming view of algorithms, and how they can be separated into two main components: the logic, describing the solution that must be found, and the control, that defines the computational technique used to find such solution. This is one of the most interesting concepts in Constraint Programming, since in the general practice of computer science the two concepts are frequently not viewed separately.

Chapter 3, **Preliminaries of Syntax and Semantics**, discusses the way syntactical elements are constructed. The author uses the *extended Bacus-Naur* form to describe constructs of logic that can be employed in constraint programming. Example of concepts discussed in this chapter are equivalence relations, state transition systems, reduction rules, and resolution calculus.

Logic Programming is discussed in Chapter 4. Logic programming is a programming model that uses the language of logic to specify computer programs. Here the separation between the control and logic part of a program becomes evident, a method which is frequently referred to as a procedural approach to programming. Logic programming is discussed in this chapter using its most prominent example, i.e., **Prolog**. In Prolog, logic expressions are represented by *Horn clauses* of the form

$$H \leftarrow G,$$

where H is called the *head* and G is the body of the clause. Logic programs in Prolog are comprised of a set of Horn clauses.

An operational semantic for Prolog is described, based on transitions and the concept of success and failure applied to logic programs. A declarative semantic for Prolog is also discussed, based on the logical meaning of the formulae expressed using its syntax. Finally, the chapter discusses the issues of soundness and completeness of logical programs.

In Chapter 5, Constraint Programming based on logic programming, or simply **Constraint Logic Programming** (CLP) is discussed. Clearly, this technique for solving constraint programs uses languages such as Prolog to find the solution of problems expressed as a set of constraints. In order to do this, the chapter describes a generalization of logic programming calculus, called CLP calculus. The CLP calculus uses symbols (such as true and false), atomic expressions (a predicate function applied to a set of terms), and constants. These are elements that form clauses of the type $A \leftarrow G$. Finally, a CP program is defined as a set of clauses. As in the previous chapter, operations in CLP are defined in terms of operational as well as declarative semantics. Soundness and completeness issues are also discussed.

Concurrent constraint logic programming (CCLP) is the subject of Chapter 6. Here, a small historic account is given, followed by a syntactical description of the concurrent language. The most important issue in the chapter is to determine operations (goals) that can be achieved in parallel, without causing deadlocks or conflicts with other goals being processed. In CCLP, the clauses have the form $A \leftarrow C|G$, and C is called a *guard*. The chapter also discusses soundness and completeness of the the CCLP.

Chapter 7 discusses **Constraint handling rules**. The basic problem is how to solve efficiently constraint programs with very different types of constraints. A general constraint solver is useful, but can perform poorly in some situations. To allow the description and implementation of solvers tailored for special types of constraints, a language called *constraint handling rules* (CHR) was

devised. This is an extension of constraint languages that allow a constraint solver to be specified using logic programming. The syntax of CHR is presented, again with discussions about semantics, soundness and completeness. The last section in this chapter is dedicated to the concept of confluence, which guarantees that the derivation of goals will give the same result, regardless of the sequence of rules that were applied for their solution.

The second part of the book starts with Chapter 8, about **constraint systems and constraint solvers**. Given a specific set of types of constraints, a constraint system is the specification of syntactic and semantic rules for that set. According to the author

> "a constraint system states which are the constraint symbols, how they are defined, and which constraint formulae are actually used in and useful for reasoning."

Some properties of constraint systems are defined, such as completeness, satisfaction-completeness, independence of negated constraints, and strong compactness.

The chapter then goes on to define capabilities and properties of constraint solvers for a CP language. Some of the properties are, e.g., correctness, failure-preservation, and independence of variable naming. Finally, some principles of constraint-solving algorithms are discussed. For example, the operations of variable elimination, local propagation, and search are presented.

The first special type of constraint system is the **boolean algebra** B, discussed on Chapter 9. This is a very simple system, where variables have only 0 and 1 as possible values. Functions are restricted to unary and binary logic operations ($\neg$, $\vee$, $\wedge$, $\Rightarrow$, $\Leftrightarrow$) and equality. For this simple constraint language, a local-propagation constraint solver is described. Other approaches, such as generic consistency methods, and theorem proving, integer programming, and boolean unification are also mentioned. An application in circuit analysis is described.

In Chapter 10, another constraint system called **Rational Trees** is discussed. The domain of application for this system is called the *Herbrand universe*, which is a set logical formulae with a fixed universe and a fixed interpretation of function symbols. It is basically a system for representation of expressions in first-order logic. The constraint solver described for this language uses the concepts of variable elimination and unification. The solver is specified with the CHR formalism described in Chapter 7. A sample application of rational trees to program analysis is also given.

Linear polynomial equations are described in Chapter 11. This CP system is is also known in the mathematical programming community as linear programming. A number of good algorithm exist to solve linear programs, and the chapter describes how such techniques can be used from the declarative point of view. This is an interesting part of the book, since most people are used only to the algebraic aspects of linear programming.

Finite domains is a CP system where variables are constrained to be from a finite set of the integers. Expressions can be constructed from the constants 0 and 1, the binary operations + and .. (for intervals), lists of symbols, and comparison operators ($=$,$\neq$, $\leq$, etc.). The chapter describes a local-propagation constraint solver for this system (again using the CHR), and applications to puzzles and scheduling.

The last type of CP system discussed in the book are **non-linear equations**, which are a more general type of system with the real numbers as the domain. All common real functions are allowed in this system, such as $+$, $\times$, sin, exp, etc. The authors present an extension of the solver in the previous chapter to handle the non-linear equation system, and discuss some of its applications.

Part III of the book is comprised of a set of applications of constraint programming, drawn from diverse areas. Chapter 14 gives an **overview of the market** for constraint programming

applications, including the commercially available systems, such as versions of Prolog, library implementations of specific systems such as the CHR language, and application oriented software. An impressive set of examples of commercial uses is given. Chapter 15 discusses an application of CP to **optimal sender placement for wireless communication systems**. The next chapter presents the problem of **rental advising**, and how this can be implemented with Prolog. Finally, Chapter 17 gives a CP approach to solve the **university course timetabling** problem.

3 Conclusion

The Essentials of Constraint Programming is a good book, if what you are looking for is just a general introduction to the subject. The book discusses many topics that give a global view of the main issues in CP, and how they are interrelated.

However, if you need a good reference book or a textbook, you should look somewhere else. Each chapter in the Frühwirth and Abdennadher's book give just a quick introduction to the topic, a few theorems without proof and some applications of the concepts. In my view, the book seems to be immature for other uses. Maybe this was the intention of the authors, but I believe that it would be more useful to have deeper discussions of at least some of the ideas exposed.

Despite some of these problems, I believe that The Essentials of Constraint Programming can be useful to many people interested only in introductory topics. There are a fair number of examples, and the non-initiated reader will at least have a good overview of the area of constraint programming, before starting with a more comprehensive literature.

Ellen DeGeneresâ€™ unprecedented apology has not been well-received, as some accuse her of throwing her own staff under the bus. Nick Bond. bondnickbond.Â Ellenâ€™s language in the letter suggests she wasnâ€™t fully aware of the alleged workplace culture at the talk show sheâ€™s hosted for more than 2700 episodes over the past 17 years. She opens by saying she wanted the show to be a â€œplace of happinessâ€ and is â€œdisappointed to learn that this has not been the caseâ€. â€œAs weâ€™ve grown exponentially, Iâ€™ve not been able to stay on top of everything and relied on others to do their jobs as they knew Iâ€™d want them done. Clearly some didnâ€™t. That will now change and Iâ€™m committed to ensuring this does not happen again,â€ she says.