Intelligent Data Analysis

Paul E. Silvey, Cynthia L. Loiselle, Paul R. Cohen

Computer Science Technical Report 92-79

Experimental Knowledge Systems Laboratory
Department of Computer Science
University of Massachusetts
Amherst, Massachusetts 01003

Abstract

We believe the problem of automating the process of building models from empirical data is a critical issue for both Artificial Intelligence and other scientific computing researchers. Although both fields require models of the behavior of complex systems, as AI researchers we may more directly address our particular needs. AI researchers require models that let us determine the influence of design decisions and environmental factors on the performance of AI programs so as to inform the design of the next generation of intelligent agents. Our research includes the complementary projects of building a blackboard-based automated model-building assistant and analyzing the efficacy of heuristics used in function finding programs.

This research was supported by DARPA-AFOSR contract F49620-89-C-00113.

Introduction

As Artificial Intelligence researchers, our basic objective is to increase our knowledge of how and why intelligent agents work the way they do. We view intelligent behavior, whether natural or artificial, as unexplained phenomena that we wish to better understand. As computer scientists, we write and study computer programs. Many of these programs exhibit behavior that we would consider intelligent, yet we are frequently at a loss to explain exactly how they do it, or to identify particular changes in circumstances that would lead them to stop doing it. The behavior of a large complex computer program (such as a real-time expert system) is often only understood in very qualitative terms. We might know, for example, that a system's performance degrades as we add additional tasks, but we don't know by precisely how much, nor whether there are significant interaction effects with other changing external conditions.

The level of understanding we seek would enable us to determine the influence of both design decisions and environmental factors on a program's performance. We know whether an increased runtime is due to, say, restructuring the knowledge base (an architectural change) or an increase in the amount of data it must handle (an environmental change). Such an understanding would allow us to *predict and analyze the behavior of AI programs.* Because such an understanding informs design, it would let us build better programs in the next iteration. Central to our ability to obtain this level of understanding is the existence of models that describe these influences.

We can attempt to build these models analytically, using techniques such as complexity analysis. Such models may or may not be relevant to running programs, however. For example, a worst-case analysis may not be useful in program design if the scenario that gives rise to such a state is unlikely in the extreme or trivially avoided. Furthermore, analytic models often require simplifying assumptions in order to make them tractable. Ascertaining whether these assumptions are justified may not be possible without implementation.

An alternative is to construct models of program behavior by observing the program itself. This ensures both that our models will be relevant to our systems, as implemented, and that any simplifications will be borne out by program behavior.

We are interested in improving our scientific understanding of a computer program's behavior through empirical analysis of its observable characteristics. In order to facilitate this, we are developing knowledge-based tools for data analysis; to help automate the process of finding interesting patterns in experimental data sets and to help us develop models that capture significant behavioral properties of our programs. The general models we hope to develop consist of causal, functional relationships between observable characteristics of a program's architecture, its environment, and its behavior.

We believe the problem of automating the process of building models from empirical data to be one of the critical issues for both Artificial Intelligence and scientific computing—for AI because we must have such models of our programs to inform the design of the next generation of intelligent agents, and by extension, for scientific computing because many fields require models of the behavior of complex systems.

We seek to understand the process of incrementally building models from data, whether by human researchers or by machines, and are designing a computer-based tool to both assist and investigate that process. As an automated assistant this work represents the development of an enabling technology for more principled Artificial Intelligence research that bases the analysis and design of AI systems on predictive models. As an investigative tool it allows us to study the application of AI techniques to the experimentation and analysis phases of scientific research.

In the next two sections we outline our research plans for constructing a blackboard-based automated model building assistant and for investigating the efficacy of existing techniques for the empirical discovery of functional relationships. The last section describes related research from the Experimental Knowledge Systems Laboratory.

An Automated Model-Building Assistant

Overview

Our basic approach to model formulation is to integrate the complementary strategies of exploratory and confirmatory data analysis (e.g., [Tukey, 1977, Wickens, 1989]) in a knowledge-based decision aid. Combining these approaches in a single system allows us to iteratively characterize subsets of a user-provided data set, detect and identify dependency relationships, and synthesize and evaluate complex models from simpler elements.

A model produced by such an automated decision-aid will consist of a set of hypothesized relationships between sets of variable data terms. Model terms are identified as observed attributes in the input data set, or they are derived from other terms through combination, transformation, or data set partitioning. Relationships between terms in the model can be described in multiple ways: as causal dependencies, qualitative trends, functional forms, or precise functional relationships. The model-building process uses both data-driven and goal-driven mechanisms in the search for models that characterize the data well, and alternate methods may exist for deriving similar results at each step. For example, several researchers have developed alternate techniques for inferring causal structure from data (e.g.,[Glymour *et al.*, 1987, Pearl and Verma, 1991]).

Frequently, the model-building process begins in a data-driven manner. A correlation coefficient or other pair-wise measure of association might be used to identify potential bivariate dependencies, or the presence of a multi-modal univariate distribution could suggest the utility of data set partitioning. At higher levels of abstraction, bivariate relationships may suggest term combination alternatives or identify probable elements of multivariate relationships.

At the same time, goal-driven processing provides a useful means to constrain the search for terms and relationships. Attention may be focused on terms deemed relevant by the human experimenter based on their experimental roles as dependent or independent variables. Information about the kinds of models we expect to find, whether provided by the experimenter or by the system's initial examination of the data, will similarly guide the application of lower level components. If we suspect that the most plausible model is non-linear, then we may favor deriving terms that produce non-linear behavior over those that maintain linear relationships. High-level techniques such as path analysis can be used in the confirmatory stage to evaluate candidate models.

Decision Aiding

Our long-term objective is to evolve to more fully automated model-building and discovery mechanisms driven by an opportunistic control strategy. However, we believe it is prudent to first develop a system that facilitates model building from data, but requires the active involvement of a human analyst. This approach also allows us to incorporate techniques for scientific visualization, and to consider important issues of human-computer interaction. We expect to develop automated strategies from our experience with the system as a manual analysis decision aid, letting users provide much of the initial reasoning control strategy. Although we are developing this system in the domain of modeling complex program behavior, we believe useful analytic strategies can be developed that apply to more general scientific modeling problems.

Architecture

The combination of multiple levels of abstraction, alternate methods for deriving intermediate results, and opportunistic reasoning, motivates our decision to implement the system using a blackboard architecture[Nii, 1986]. A shared blackboard data structure holds elements of the developing model or models, and the individual knowledge sources operate on those elements to create terms, hypothesize relationships, fill-in attribute slots, or perform statistical tests. Blackboard levels reflect the space of hypotheses explored during the process of model building. They include the raw data table, univariate terms, bivariate relationships, multivariate model components, and causal linear path models.

Data analysis and model construction requires a rich set of analytic tools. Many of these "tool-kit" knowledge sources compute important statistics for describing terms and relationships. Examples include knowledge sources that characterize univariate distributions by their statistical moments, identify outliers in the raw data set, and compute measures of association such as linear correlation. Higher level analytic knowledge sources include ones that assess non-linearity in bivariate relationships, partition the data set into classes, create new terms through arithmetic combination, perform multivariate regression, and evaluate path-analytic models.

But more importantly, the combinatorics of exploring the space of possible models requires a strategic control component. In addition to external guidance provided by the human analyst, there are control knowledge sources to suggest which terms to combine into new terms, which operators to apply to those terms, and where the system's focus of attention should be at various blackboard levels. The blackboard architecture allows us to incorporate both analytic tool-kit knowledge and strategic control knowledge in a modular and incremental fashion. This modularity in turn provides us with the ability to evaluate the relative contributions of various knowledge sources through ablation studies, thus utilizing the methodology for which the tool is being developed to better understand the tool itself. (See the section on Previous and Related Research for more on this methodology.)

Modeling and Empirical Discovery

Researchers in the machine learning subfield of empirical discovery have investigated the problem of deriving models to capture relationships in a given data set. When such models take the form of mathematical expressions, we usually refer to the problem as one of *function finding*. In service of developing the tool-kit and control knowledge sources for the model building assistant described in the previous section, we have begun to investigate the utility of existing model building techniques found in such function finding programs.

Most existing function finding programs (e.g., [Langley *et al.*, 1983, Falkenhainer and Michalski, 1986, Żytkow, 1987, Zembowicz and Żytkow, 1992]) combine a search strategy with an evaluation heuristic to search the space of potential models. [1] For example, IDS [Nordhausen and Langley, 1990] uses beam search on nodes ranked by the correlation of the model's computed values with the known dependent values from the data set. The search space is, of course, defined by the set of term creation operators chosen for the particular system. Term creation operators are cho-

[1] For a notable exception see [Schaffer, 1991] which suggests that function finding might be more properly viewed as a classification problem.

sen to reflect the kinds of models we expect to find—if we anticipate that data will be well modeled by linear relationships then our operators create new terms accordingly.

The effectiveness of any function finding technique depends, therefore, on the ability of the chosen search strategy and evaluation heuristic to accurately reflect the search space's terrain. Said more simply, different function finding techniques will be more or less effective in discovering different kinds of models. We view this terrain, the space of models that may be considered, to be the environment of the function finding algorithm. As with any AI program, we seek to understand function finding programs in terms of how architectural components (the search strategy and evaluation heuristic) operate in specific environments (the model space) to produce behavior (the effectiveness of the algorithm in discovering the underlying relationships).

The goal of our function finding research is to identify effective heuristics for automated model building. We believe these heuristics fall into two classes. First are those that suggest specific classes of models and therefore identify the appropriate term creation operators. This can be seen most clearly in the case of bivariate relationships where the shape of a curve can suggest appropriate models. In addition to the familiar linear relationship, superlinear and sublinear curves suggest various exponential, reciprocal and logarithmic relationships (e.g., a superlinear curve where y increases as x increases might suggest either a power function $y = \alpha x^{\beta}$ where β is a constant > 1, or an exponential form $y = \alpha e^{\beta x}$ where α and β are constants, with $\beta > 0$).

Second are those heuristics that evaluate terms as potential (possibly partial) models. Recalling the characterization of function finding as search, these heuristics evaluate the promise of a node in the search space in terms of its likelihood to lead to a solution, an acceptable model. Evaluating the effectiveness of these heuristics must be done in the context of the search strategy in which they are embedded and the characteristics of the model space we wish to search. We wish to build models of computer programs, specifically AI programs. Thus we must perform this evaluation in the environment corresponding to the space of models that may describe the behavior of AI programs.

To this end we are building a highly parameterized, modular function finding system that will allow us to investigate the effectiveness of heuristics from the latter class. Within this system we may select a function finding technique by choosing from a set of search strategies (e.g., exhaustive search, beam search, A*) and evaluation heuristics (correlation, equality, goodness of fit measures). Similarly, sets of term creation operators (addition, multiplication, division, transformations, etc.) will be chosen to reflect different model spaces. Given a target model and corresponding data set, each such instantiation may then be used to map the terrain of the corresponding search space as viewed by this function finding technique (combination of search strategy and evaluation heuristic). This will allow us to evaluate the ability of these evaluation heuristics to guide search in the context of different search strategies and model spaces.

For example, if correlation is used to reflect the promise of a node in terms of estimated distance to a solution, then we expect that states in the search space closer to the solution should yield higher correlations than those father away. Said more precisely, such states should correspond to candidate models that predict values more highly correlated with those produced by the target model, than those predicted by candidate models corresponding to states farther from the solution. A simple examination of the corresponding correlations for specific target models and data sets will let us know if this expectation is borne out.

But since distance to the goal is used differently by different search strategies, the results of the above examination may be more or less relevant to those specific strategies. If correlation is found to correspond well with proximity to the solution in most cases, but occasionally far flung nodes also display surprisingly high correlations, then best first search, by focusing on the single most highly correlated candidate model, may be led astray into pursuing false leads, but beam search, which considers the top *set* of promising nodes, may be less susceptible to this effect.

We believe that understanding the effect of different search strategies and model spaces on the efficacy of search evaluation heuristics will allow us to construct appropriate function finding techniques more finely tuned to the expectations we have about the nature of the underlying model. The knowledge we gain from these experiments will be used to inform the corresponding knowledge sources in our automated model-building assistant.

Previous and Related Research

The Experimental Knowledge Systems Laboratory (EKSL) performs research in the design of autonomous agents for complex, real-time environments. The need for a principled approach to this problem has led to the development of the Modeling, Analysis and Design (MAD) methodology [Cohen, 1991], an approach we believe bridges the gap between theoretical and systems-oriented AI research. MAD combines analytical modeling with empirical substantiation to justify the agent architectures that we use. EKSL has developed a number of tools that support model-building and analysis, as well as the Phoenix system [Cohen *et al.*, 1989], a testbed environment for this research. These tools include statistical analysis and data manipulation packages, several analytic techniques, and a prototype model-building decision aid. This prototype was developed to evaluate graphical user interface techniques for manual term creation

and assessment, and for incremental model visualization. It also provided the basis for the design of the blackboard-based model building assistant described above.

The Common Lisp Analytical Statistical Package (CLASP) [Fisher, 1990] was developed for the statistical analysis of large data sets. This modularized system can be used as an interactive analysis tool, providing powerful data manipulation tools, many of the most common parametric and non-parametric statistical tests, and plotting capabilities. It can also be accessed as a runtime library by programs (e.g., Phoenix agents) using statistical and probabilistic models. CLASP manipulates data using the Relational Table Manager (RTM), a relational database management tool we developed which provides type- and consistency-checking as well as the powerful operations of relational algebra.

EKSL has worked extensively in analyzing the behavior of complex computer programs. Howe [1992, 1992] developed the procedure of *failure recovery analysis*, a method of analyzing execution traces to discover when a planner's actions may be causing failures, and recommending redesigns to avoid or mitigate the failures. Critical to this technique is the statistical analysis of execution data for dependencies between actions and failures. Hart and Cohen [1992] examined the environmental and architectural factors affecting the performance of the Phoenix planner, testing predictions about the planner's robustness against variations in some of these factors. This analysis employed the statistical technique of *path analysis* [Li, 1975] for constructing and testing causal explanations of the planner's behavior. Path analysis is a useful technique for constructing the kinds of linear path models envisioned for our system.

Tools such as these provide the base capabilities for our automated model-building assistant. Initial knowledge sources can draw on our statistical tools, such as CLASP's path analysis module, as well as our experience in analyzing the performance of the Phoenix planner.

Acknowledgments

We wish to thank Dave Hart for many thoughtful comments on drafts of this paper. The terrain mapping research proposed in the section on Modeling and Empricial Discovery is based on an idea from Eric Hansen. This research was supported by a DARPA-AFOSR contract, F49620-89-C-00113. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

References

Cohen, Paul R.; Greenberg, Michael L.; Hart, David M.; and Howe, Adele E. 1989. Trial by fire: Understanding the design requirements for agents in complex environments. *AI Magazine* 10(3):32–48.

Cohen, Paul R. 1991. A survey of the Eighth National Conference on Artificial Intelligence: Pulling together or pulling apart? *AI Magazine* 12(1):17–41.

Falkenhainer, Brian C. and Michalski, Ryszard S. 1986. Integrating quantitative and qualitative discovery: The ABACUS system. *Machine Learning* 1:367–401.

Fisher, David E. 1990. Common Lisp Analytical Statistics Package (CLASP). Technical Report 90–15, Computer Science Department, University of Massachusetts, Amherst, MA.

Glymour, C.; Scheines, R.; Spirtes, P.; and Kelly, K. 1987. *Discovering Causal Structure: Artificial Intelligence, Philosophy of Science, and Statistical Modeling.* Academic Press, Orlando, FL.

Hart, David M. and Cohen, Paul R. 1992. Predicting and explaining success and task duration in the phoenix planner. In *Proceedings of the First Internation Conference on AI Planning Systems*, College Park, MD. Morgan Kaufmann Publishers, Inc. 106–115.

Howe, Adele E. and Cohen, Paul R. 1992. Debugging plan failures by analyzing execution traces. Technical Report 92-22, Computer Science Department, University of Massachusetts, Amherst, MA.

Howe, Adele E. 1992. Analyzing failure recovery to improve planner design. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, San Jose, CA.

Langley, Pat; Bradshaw, Gary L.; and Simon, Herbert A. 1983. Rediscovering chemistry with the BACON system. In Michalski, Ryszand S.; Carbonell, Jaime G.; and Mitchell, Tom M., editors 1983, *Machine Learning: An Artificial Intelligence Approach.* Morgan Kaufmann Publishers, Inc., Palo Alto, Ca. chapter 10, 307–329.

Li, C. C. 1975. *Path Analysis—A Primer.* Boxwood Press.

Nii, H. P. 1986. Blackboard systems. *AI Magazine* 7(2 and 3):38–53, 82–106.

Nordhausen, Bernd and Langley, Pat 1990. A robust approach to numeric discovery. In Porter, Bruce W. and Mooney, Ray J., editors 1990, *Proceedings of the Seventh International Conference on Machine Learning*, Austin, Texas. Morgan Kaufmann Publishers, Inc. 411–418.

Pearl, J. and Verma, T. S. 1991. A theory of inferred causation. In Allen, J. A.; Fikes, R.; and Sandewall, E.; editors 1991, *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, San Mateo, CA. Morgan Kaufmann Publishers, Inc. 441–452.

Schaffer, Cullen 1991. On evaluation of domain-independent scientific function-finding systems. In Piatetsky-Shapiro, Gregory and Frawley, William J., editors 1991, *Knowledge Discovery in Databases.* AAAI Press. chapter 5, 93–104.

Tukey, John W. 1977. *Exploratory Data Analysis.* Addison-Wesley, Reading, MA.

Wickens, T. D. 1989. *Multiway Contingency Tables Analysis for the Social Sciences.* Lawrence Erlbaum Associates, Hillsdale, NJ.

Zembowicz, Robert and Żytkow, Jan M. 1992. Discovery of equations: Experimental evaluation of convergence. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, San Jose, CA. MIT Press. 70–75.

Żytkow, J. M. 1987. Combining many searches in the FAHRENHEIT discovery system. In *Proceedings of the Fourth International Workshop on Machine Learning*, Irvine, CA. 281–287.