Integrating Many Techniques for Discovering Structure in Data

Dawn E. Gregory Paul R. Cohen

Experimental Knowledge Systems Laboratory
Computer Science Department, LGRC
University of Massachusetts
Box 34610, Amherst, MA 01003-4610

Phone: (413) 545-3616
Fax: (413) 545-1249
E-mail: {gregory,cohen}@cs.umass.edu

Abstract. This paper describes a formal representation of the discovery process that efficiently integrates of any number of data analysis strategies, regardless of their similarities and differences. We have implemented a system based on this formalization, called the *Scientist's Empirical Assistant* (SEA). SEA employs several analysis strategies from the discovery literature, including techniques for function finding, causal modeling, and Bayesian conditioning. It uses high-level knowledge about the discovery process, the strategies, and the domain of study to coordinate the selection and application of analyses. It relies on the skills and initiatives of an expert user to guide its search for structure. Finally, it designs and runs experiments with a simulator to verify its findings.
SEA is currently capable of performing a full cycle of discovery and analysis, from hypothesis formulation to experiment design, data collection, analysis, and the generation of explanatory hypotheses. In addition, the formalization on which it is based provides an environment for studying the how, what, why, and when of intelligent data analysis.

Integrating Many Techniques for Discovering Structure in Data

Abstract. This paper describes a formal representation of the discovery process that efficiently integrates of any number of data analysis strategies, regardless of their similarities and differences. We have implemented a system based on this formalization, called the *Scientist's Empirical Assistant* (SEA). SEA employs several analysis strategies from the discovery literature, including techniques for function finding, causal modeling, and Bayesian conditioning. It uses high-level knowledge about the discovery process, the strategies, and the domain of study to coordinate the selection and application of analyses. It relies on the skills and initiatives of an expert user to guide its search for structure. Finally, it designs and runs experiments with a simulator to verify its findings.
SEA is currently capable of performing a full cycle of discovery and analysis, from hypothesis formulation to experiment design, data collection, analysis, and the generation of explanatory hypotheses. In addition, the formalization on which it is based provides an environment for studying the how, what, why, and when of intelligent data analysis.

1 Motivation

Data analysis is not simply a matter of applying a formula to a set of data: it is a complex process that involves selecting an appropriate representation, designing an analysis strategy, gathering data into an appropriate format, applying the formula(e), and explaining the results. Thus, if we want data analysis programs that behave intelligently, they must be designed to accomodate each of these tasks.

Recent research in Machine Learning and Scientific Discovery indicates that systems with flexible analysis strategies and a variety of representations are more proficient at uncovering structure in data. There are several reasons to suspect this is true. First, in having several techniques to choose from, a system can select the one most suited to the analysis problem at hand (e.g. [2]). Second, integrated systems have the advantage of supplementing one kind of result with others, information which often constitutes an *explanation* of previous findings (e.g. [7]). Finally, interesting structure is often discovered in the process of shifting from one representation to another [10]. Thus, there is a significant advantage to integrated discovery systems.

This paper describes our approach to the integration of analysis techniques and a system, called the *Scientist's Empirical Assistant* (SEA), that implements this view. Section 2 presents an example of complex data analysis that employs a variety of strategies and representations. We then make several high-level observations about the analysis process to motivate our design, which is described in section 4. Finally, we describe the key contributions and identify areas of future work.

2 Example

We begin with a detailed example of data analysis to show how different strategies and representations might be employed to discover structure within data. This analysis was originally performed manually, as described in [5], and has subsequently been replicated by SEA under the guidance of a human user.

The example focuses on a real-world problem from the domain of parallel computing. Parallel architectures employ several processing units that work in conjunction to solve problems more quickly than is possible on a single processor. In theory, P processors should reduce the overall running time by a factor of P; for example, N unit-time tasks should be executed in time $T = N/P$. In practice, connectivity among the processors and interdepencies among the tasks have a significant impact on performance, because some processors may be forced to remain idle for long periods of time. To deal with this problem, designers are highly concerned with policies for *balancing* the load among processors, to ensure that idle periods are a brief as possible.

In this problem, data analysis is used to discover the properties of two load-balancing policies for a particular parallel architecture on a specific kind of computing task.[1] Simulations of the architecture and stochastically-generated input problems generate the data used in analysis. The simulation is configured by three parameters: the number of processors, P, the load-balancing policy, Π, and an input parameter, α. Among the values output by the simulator are the net running time, T, and the number of unit-sized tasks, N. Simulation of both policies on architectures of different sizes and various input settings yielded a preliminary dataset of 360 data points containing the variables Π, P, α, N, and T.

The purpose of the analysis is to determine whether optimal performance is achieved; that is, we want to determine if $T = N/P$. Given the preliminary dataset, the first strategy is to try a *t-test* on the paired samples of T and N/P. The test indicates that T and N/P are not significantly different, which lends support to the hypothesis but cannot be considered conclusive evidence (i.e. the test cannot *accept* the null hypothesis $T = N/P$). Thus we must consider additional strategies for testing the hypothesis.

A simple manipulation of data suggests a new strategy: if $T = N/P$, the difference $\epsilon = T - N/P$ should be approximately 0. At first glance it appears we are simply replicating the first analysis, but the variance of ϵ is much smaller than that of T, making significant results more likely. Indeed, a one-sample *t-test* whether ϵ has mean 0 yields a significant result, *rejecting* the hypothesis that $\epsilon = 0$ and concluding that $\epsilon > 0$. This finding contradicts the initial hypothesis, so we look for factors that predict the value of ϵ.

To decide which factors influence ϵ, it is necessary to bring in a new representation for the relationship between ϵ and other variables. In this case, we are interested in *dependencies* among variables; for example, whether the value of

[1] Detailed descriptions of the architecture, the task, and the policies are irrelevant to the ensuing discussion, so we refer the interested reader to [5] for more information.

ϵ depends on P. Heuristic rules generate hypotheses about potential dependencies: ϵ might depend on any of P, Π, α, or N.[2]

The new representation indicates a different set of analysis strategies which are applied in turn. P, Π, and α are independent variables and can thus be considered discrete-valued for analysis, so one-factor analysis of variance (ANOVA) is employed. N is strictly numeric so its influence is tested with linear regression. From these analyses, we find that P, Π, and N all have significant effects on ϵ, but α does not.

At this point, there are several possibilities for further analysis. We can use two-factor ANOVA to determine whether there is an interaction between P and Π. We can explore the separate effects of P or Π on the relationship between N and ϵ. Or, we might skip both of these tasks and immediately start exploring the combined effects of P and Π on the relationship between N and ϵ.

Regardless of the approach initially chosen, we eventually arrive at several interesting results: the slope of the regression of ϵ on N increases as P increases; the slope is consistently larger for one of the policies given each value of P; the slope is non-zero for some combinations of P and Π, but not all. These results can be explained by external knowledge about the load-balancing problem. As P increases, it is more likely that some processors will become idle during the computation, regardless of the load-balancing policy employed. One policy does a better job at balancing the load, and this policy apparently behaves optimally (i.e. the slope is approximately 0) for some values of P. These results are quite useful to the designer, who now has empirical evidence that a certain configuration will utilize its resources effectively.

3 Observations

The example described above has several important features that are common to data analysis. First, it shows that analysis is not simply a matter of applying a formula to obtain a result — it is an *iterative* process that relies heavily on context and previous results to identify which formula is appropriate and how the results should be interpreted. Second, the goal of data analysis is to develop a model that accurately predicts the values of variables, and this goal is attained through *variance reduction* techniques. Finally, the model should correspond to the true structure of the world, so its predictions must be reconciled with higher-level knowledge of the domain. Often, one or more of these features is overlooked by the designers of data analysis systems.

3.1 A Complex, Knowledge-Intensive Process

Data analysis is an iterative process consisting of several stages. First, the question to be addressed by analysis is formalized as a *hypothesis*; for example, the question of whether an architecture performs optimally is initially represented

[2] By definition ϵ depends on T, so this hypothesis is not generated.

by the hypothesis $T = N/P$. The form of the hypothesis produces *strategies* for analysis: the "=" in $T = N/P$ indicates that we might try a t-test. Next, *observations* of real behavior are collected and arranged in the required format. The analysis strategy supplies a formula that is applied to the observations, yielding a *result*. Finally, the result must be explained by considering the *reasons* it may occur; for example, $T = N/P$ can be explained by $\epsilon = 0$.[3] The task of explanation often generates new hypotheses, and the whole process repeats.

Every data analysis system must consider each of these stages, but none has automated all of them. Statistical packages provide facilities for hypothesis testing, but must rely on the user to formulate hypotheses, select the appropriate analysis, prepare the data, and explain the results. Intelligent discovery systems, such as Bacon [6], Tetrad [4], and C4.5 [8], formulate hypotheses of a specific form and then perform analysis in this context, but rarely have facilities for gathering and preparing data or explaining the results of analysis. Other discovery systems have addressed these latter concerns in the context of certain domains; for example, Fahrenheit [11] deals with issues of experiment design within the domain of chemistry. Unfortunately, none of these systems supports all the stages in a flexible, domain-independent manner.

There are good reasons why intelligent data analysis has been restricted to certain domains or specific types of hypothesis. First, each stage of the process may produce several potential courses of action, and selecting among these alternatives is a nontrivial task. By focusing on a specific domain or hypothesis test, it is possible to minimize the number of choices and to formalize policies for decision-making. Second, experiment design and data collection are difficult to automate, because they rely heavily on domain knowledge and the ability to interface with the physical world. Again, restricting the context makes automation possible. Finally, explaining results requires a deep semantic knowledge of the domain, the source of the data, the underlying assumptions, and valid interpretations of the analysis. In sum, it is the complexity of the analysis process and the need for high-level knowledge that restricts the cabilities of data analysis systems.

3.2 Reduction of Variance

The desired outcome of analysis is a formal model of behavior that yields accurate predictions about the values of variables. Prediction accuracy is measured by *variance*, the difference between predictions and actual observations. Data analysis is a mechanism for explaining variance, to identify which factors contribute to it. When successful, analysis indicates that a model will continue making accurate predictions; on failure, it means that the model should be modified to account for more of the variance. Thus, variance reduction is a major principle behind intelligent data analysis.

[3] The reader may question whether $\epsilon = 0$ is truly an *explanation* of the finding $T = N/P$. However, it is not the hypothesis $\epsilon = 0$ itself, but the subsequent justification of this hypothesis that constitutes an explanation.

Let us consider how this variance reduction principle takes form in some more common analysis strategies. Analysis of variance is a parametric technique that decides whether a categorical variable X influences the value of numeric variable Y. This decision is directly based on the variance reduction principle: if the variance of Y is significantly reduced by accounting for X, then X influences Y. The t-test decides whether two values can be considered equal given the background variance; significant results are more likely when variance is reduced. Linear regression determines if a linear relationship exists between two numeric variables; again, smaller variance makes a significant result more likely. All of these techniques are based on the *generalized linear model*, which decomposes net variance into a sum of effects for each factor, interactions between factors, and underlying background variance.

Other techniques that do not explicitly incorporate statistical variance still make use of this principle. Bayesian conditioning, for example, compares the *prior* probability of an event A, given only background knowledge, with its *posterior* probability given a specific condition B. When the posterior probability is large compared to the prior, we conclude that B is useful for predicting A, thereby reducing the prediction error, or variance.

3.3 Identifying True Structure

Models that make accurate predictions are desirable because, in principle, the most accurate model is one that captures the true structure of the world. In practice, this is not always the case: background variance, missing or censored data, measurement error, and faulty conclusions may lead to an accurate model that has little to do with the mechanisms that generate behavior. Thus, it is important to distinguish real effects from those that appear (or do not appear) in the data.

Unfortunately, the problem of deciding whether an effect is real or not may never be resolved. This is due to a problem we call the *limit of perspective*: in order to perform analysis, a specific perspective on the problem must be established, and any such perspective is necessarily limited in its ability to ascertain the truth.

With respect to data analysis, there are three such limiting factors. First, analysis is limited to the observations that are gathered and evaluated. Since it is not feasible to record all possible observations, the range of behaviors captured in data is restricted. Of course, through random sampling we can *assume* that all interesting behaviors will be reflected in the data, but we cannot *guarantee* this is so.

Second is the notorious *latent variable problem*: analysis focuses on a specific set of variables, and we cannot be sure that all relevant variables have been included in the set. Thus, it is desirable to be able to incorporate new variables as analysis proceeds, an event that often leads to deeper understanding of behavior. For example, consider the advances following the discovery of the "atom" in chemistry or the "tectonic plates" in geology.

Finally, the models themselves impose a specific, limited perspective from which to view the data. For example, the model $T = N/P$ depicts equivalence between the values of T and N/P, but it does not consider any underlying causal relationships or the precedence among variables. On the other hand, a causal model describes direct, causal influences, but does not expose the mechanism responsible for these effects. Thus, any type of model is restricted in the reality it can represent.

These problems lead to the observation that no level of data analysis can identify the true structure in data. That being the case, how is it that data analysis has become a widely accepted approach to scientific modeling? It is because analysis is only part of the process: the real power lies in the correspondence between analysis and theory, the mapping between the semantics of the model and the mechanisms underlying behavior. In establishing and justifying such relationships, data analysis leads to a sound theory grounded in the real world.

4 The Scientist's Empirical Assistant

We have incorporated these principles into an intelligent discovery agent, the Scientist's Empirical Assistant (SEA). The design takes a meta-perspective on the problem of automated data analysis, isolating the domain- and task- specific issues from the high-level concerns discussed above.

To circumvent some of the trickier problems with the discovery process, we make two simplifying assumptions. First, SEA works in conjunction with a human user who is an expert in the domain of study and can supply external domain knowledge, make strategic decisions, and establish exploration goals. Because human scientists rarely work in complete isolation, it is reasonable that our fledgling scientist should rely on the experience and knowledge of an expert. Second, SEA assumes that empirical observations are generated by a simulator, so it can design and run its own experiments on-line during analysis. Again, this is a reasonable assumption because simulations are intended to reflect the underlying processes and relevant properties of their real-world counterparts.

SEA is a working system that has been used to replicate the study described in section 2. Given an initial hypothesis input by the user, it defines an analysis strategy, collects experiment data, performs statistical tests, and explains the results of analysis, often by generating new hypotheses.

4.1 Control Architecture

SEA accomodates the iterative, multi-stage process of data analysis through partial-hierarchical planning (PHP) [3]. *Planning* is the problem of selecting a sequence of actions that move the system from its current state to some desired, or *goal*, state. SEA has the high-level goal of deriving a predictive model, and this goal takes different forms depending on the stage of analysis and the contents of the model. For example, when the model is empty, the goal is to define some variables and generate predictive statements about their behavior; when

a hypothesis is generated, the goal is to test it against empirical data. Because the goals can change so frequently, the planner is *reactive*, which means that it executes actions while it is planning.

Actions in PHP are structured control constructs called partial plans, or simply *plans*. Each plan satisfies a specific goal that moves the system closer to its desired state. In addition, plans may post new goals, which in turn trigger some set of matching plans. This results in a hierarchy of goals which induces a hierarchy of potential actions.

A plan contains a procedural representation of the steps in an action, at either an abstract or explicit level. For example, the plan for a t-test is to compute means and variances from data, apply the formula for the t statistic, lookup the probability of obtaining that value, and reach a conclusion based on the probability. Plans may contain iteration, conditionals, executable statements, variable bindings, and subgoals; thus, the planning language is essentially a high-level programming language.

The planner used by SEA was developed by St. Amant [9] and applied to the problem of exploratory data analysis. In addition to the basic PHP mechanisms mentioned above, the planner includes two important features. First, it can be easily extended to accomodate new control constructs in the planning language. More importantly, the planner provides a facility for managing the decision points in planning through a *meta-planner*. Whenever a goal can be satisfied by multiple plans, the planner creates a *focus point* to keep track of the alternatives. At any time, SEA can abandon its current course of action in favor of another alternative from any existing focus point. The meta-planner manages such decisions, using its own set of plans to decide which focus point to select and which alternative should be pursued.

The user's strategic expertise is also incorporated through the focusing mechanism. Whenever a focus point is encountered, the absence of relevant meta-plans indicates that the user should decide which option to pursue. As with the meta-planner, the user can interrupt processing at any time to select an alternative strategy. Thus, the user is sometimes treated as the "default meta-planner".

4.2 Data structures

SEA has plans that implement the many tasks of data analysis, such as hypothesis generation, analysis planning, experiment design, statistical computations, and explanation of results. Each of these tasks involves manipulation and transformation of certain data structures; for example, analysis planning involves a transformation from a hypothesis to an analysis strategy. Because the tasks are distinguished at a high level by the data structures they work with, we briefly describe the primary data structures and their place in analysis.

Variables represent features of the world that are considered during analysis. In some sense, variables are a bridge between the symbolic world of the model and the value-based world of observations. As such, the variable data structure carries information that relates its symbol to its values. For example, each variable represents data of a certain type, such as categorical, boolean, or numeric. With

respect to experiment design, it is also important to know whether the variable represents a parameter that can be externally controlled, or a measurement that depends on behavior. This type of information determines the role of each variable in modeling; for example, it is not necessary to predict the value of controlled variables.

Variables are used to formulate *statements* about behavior. Statements are contentful, structured expressions that make predictions about the values that will be observed. Some of the statements currently supported by SEA include dependency and causation, equality and inequality, and the effects of treatment variables. Statements constitute the domain knowledge SEA manipulates during analysis; some statements are treated as *assumptions* provided by the user, while others are *research questions* to be addressed by data analysis.

Hypotheses are empirical questions that can be tested directly. The difference between statements and hypotheses is subtle, because the tasks of prediction and evaluation are highly interdependent. However, the distinction is useful for reasoning about data analysis, because the implicit mapping from predictive statement to hypothesis test is made explicit. In our formulation, statements cannot be tested directly, only through verification of supporting hypotheses.

SEA uses heuristic rules to generate hypotheses that reflect the *implications* of a statement; for example, the statement $T = N/P$ has several implications: both N and P affect the value of T (dependency); the value of T is equivalent to N/P (equality); and the error $\epsilon = T - N/P$ is zero (constant difference). Each such hypothesis is then tested, and the conclusion is used to verify or reject the original statement.

Each hypothesis has one or more *analysis strategies* that will produce a conclusion about its validity. Analyses often implement statistical hypothesis tests, but they may also be based on deductive techniques, computer-intensive resampling, or prospective experimentation. Each result derived through analysis must be explained, a task which often generates new hypotheses.

Statistical analyses rely on observational data that is accessed by generating a *data request*. The data request contains a set of variables and values for independent variables, as well as sampling assumptions the data should conform to (e.g. whether samples are independent or steps in a time series). Data requests extract the desired data from available *experiment datasets*. If no applicable data can be found, the data request will generate an *experiment plan* which is used to collect a new dataset with the specified parameters. Experiment plans are protocols for data collection, specifying sample values for independent variables and rules for taking observations. Experiment plans rely on the instrumentation package CLIP [1] to run the experiment and collect data.

4.3 Meta-Level Strategy

The planner selects and applies plans to manipulate the primary data structures according to a high-level plan for data analysis. This high-level plan alternately creates new data structures and then posts a goal to gather relevant supporting information; for example, variables are supported by predictive statements about

their values, and analyses are supported by experiment data. An outline of this process is depicted in figure 1.

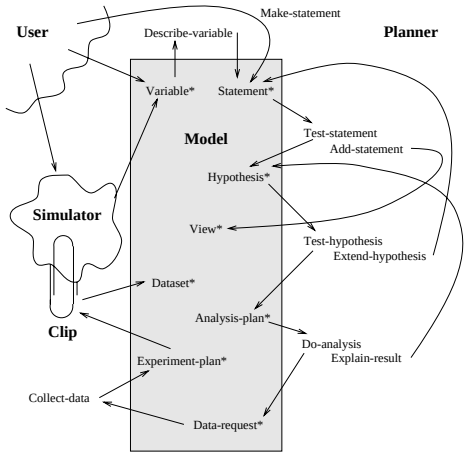

Fig. 1. An overview SEA's data analysis strategy.

Unfortunately, data analysis is not a purely algorithmic task, so this control strategy must be modified to accomodate the sometimes exploratory nature of data analysis. If we view the control strategy as a search process, the basic algorithm is a depth-first search. We can modify the search by identifying rules for selecting the next node to expand; these rules are implemented as *meta-plans*. Meta plans select a new plan (search node) based on the current state of the system, including the current goals.

5 Conclusions and Future Work

SEA integrates a variety of analysis strategies for different representations. Integrating representations and their various strategies is an important problem in intelligent data analysis.

SEA incorporates much-needed external knowledge, both in terms of domain facts and efficient strategies. Techniques for incorporating domain knowledge are encoded in the plans, inducing a tradeoff between generality and expressive power. Because the purpose of domain knowledge is primarily expressive power, we are not terribly concerned with the loss of generality. However, we would like to note that the procedural nature of the plans provides all the generality that is needed: data analysis is a *process*, and plans are a general, well-defined representation for processes.

SEA provides an opportunity for studying the high-level strategies employed by expert scientists. These can be addressed formally, by symbolically encoding

these strategies as meta-plans, or informally, by observing the behavior of expert scientists using the system. An important area of future work is to incorporate unsupervised and/or supervised learning methods to automatically develop high-level analysis strategies through experience with an expert user.

SEA needs to be formally evaluated. First we need to evaluate its generality by trying a variety of analysis problems. Then we need to determine its proficiency as a scientific assistant by considering performance in three conditions: the user alone, SEA alone, and SEA and the user together.

SEA is a testbed for the study of analysis and discovery methods. It provides the opportunity to compare and contrast alternative strategies and/or representations. The application of these methods to real-world problems helps determine their strengths and weaknesses. It also provides the opportunity to develop new discovery heuristics, including the implications of external knowledge and meta-level strategies. This is due to its formalization of the data analysis process.

References

1. Scott D. Anderson, Adam Carlson, David L. Westbrook, David M. Hart, and Paul R. Cohen. CLASP/CLIP: Common Lisp Analytical Statistics Package/Common Lisp Instrumentation Package. Technical Report 93-55, Computer Science Department, University of Massachusetts at Amherst, 1993.
2. Carla E. Brodley. Addressing the selective superiority problem: Automatic algorithm/model class selection. In *Proceedings of the Tenth International Machine Learning Conference*, pages 17–24, 1993.
3. Michael P. Georgeff and Amy L. Lansky. Procedural knowledge. *Proceedings of the IEEE Special Issue on Knowledge Representation*, 74(10):1383–1398, 1986.
4. Clark Glymour, Richard Scheines, Peter Spirtes, and Kevin Kelly. *Discovering Causal Structure: Artificial Intelligence, Philosophy of Science, and Statistical Modeling*. Academic Press, Orlando, FL, 1987.
5. Dawn Gregory, Lixin Gao, Arnold L. Rosenberg, and Paul R. Cohen. An empirical study of dynamic scheduling on rings of processors. In *Eighth IEEE Symposium on Parallel and Distributed Processing*, 1996. To appear.
6. Pat Langley, Herbert A. Simon, Gary L. Bradshaw, and Jan M. Zytkow. *Scientific Discovery: Computational Explorations of the Creative Processes*. MIT Press, Cambridge, MA, 1987.
7. Bernd Nordhausen and Pat Langley. An integrated approach to empirical discovery. In Jeff Shrager and Pat Langley, editors, *Computational Models of Scientific Discovery and Theory Formation*. Morgan Kaufmann, 1990.
8. J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
9. Robert St. Amant and Paul R. Cohen. A planner for exploratory data analysis. In *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems*, pages 205–212. AAAI Press, 1996.
10. Raul E. Valdes-Perez. Some recent human/computer discoveries in science and what accounts for them. *AI Magazine*, 16(3):37–44, 1995.
11. Jan M. Zytkow, Jieming Zhu, and Abul Hussam. Automated discovery in a chemistry laboratory. In *Proceedings of the 8th National Conference on Artificial Intelligence (AAAI-90)*, pages 889–894, 1990.

This article was processed using the LaTeX macro package with LLNCS style