AX: SEARCHING FOR DATABASE REGULARITIES USING CONCEPT NETWORKS

Donald J. Berndt
Department of Information Systems
Stern School of Business
New York University

Working Paper Series
STERN IS-95-11

AX: Searching for Database Regularities Using Concept Networks

Donald J. Berndt
Information Systems Department
Stern School of Business, New York University
44 West 4th Street, New York, NY 10012-1126
dberndt@stern.nyu.edu

July 18, 1995

Abstract

In many organizations, both business and scientific, we collect ever increasing amounts of data using information technology. Indeed, the technology for collecting data has outpaced our ability to analyze and interpret these very large databases. In this paper, we discuss the interaction of heuristic search and domain knowledge in the AX knowledge discovery tool. The search process rests on the use of rule quality measures and the organization of domain knowledge. A small loan application database from the machine learning repository is used to illustrate the process.

Keywords: knowledge discovery, machine discovery, domain knowledge, heuristic search.

1 Introduction

In many organizations, both business and scientific, we collect ever increasing amounts of data using information technology. Whether you do business at a bank, a supermarket, or on the stock exchange, an electronic footprint of your passage is left in information space. Overall, it has been estimated that the world data supply doubles every 20 months [FPSM91]. Indeed, the technology for collecting data has outpaced our ability to analyze and interpret these very large databases. Streams of information such as credit card transactions, retail product purchases, and economic data are producing massive pools of information that should yield benefits relative to the expense of collection. How can we deal with these rapidly increasing volumes of data? We do not envision users directly navigating through mind-numbing databases, rather we see them employing tools that facilitate the crystallization of useful patterns or knowledge. To facilitate this, we need to design computer tools that support the analysis of very large databases.

In this paper, we outline the design of the prototype AX discovery system. This tool is intended to test some specific discovery techniques as well as provide a flexible infrastructure for future experiments. For this reason, the natural fit between databases and logic programming made Prolog an interesting language for the implementation. Discovered knowledge is represented as rules within the system and some rule quality measures are discussed in Section 2. In particular, we compliment the usual conditional probability measure with a related measure we call *power*, as well as rule scope or database selectivity. The system is organized around the notion of *concept sets*, which contain database attributes or views. Rule discovery consists of applying various techniques to uncover relationships between individual concepts. New concept sets can be constructed by the user or generated as part of the heuristic search process. The search process relies on domain knowledge encoded as *concept networks*—generalizations of attribute hierarchies (for instance, see [HCC92]). Currently, rule discovery within a given concept set is based on an implementation of abstract-driven discovery as outlined in [DT93], see Section 4.1.

1.1 Knowledge Discovery Tools

Some of the large business and scientific databases are carefully analyzed by hand. However, detailed data analysis can be time consuming and probably occurs infrequently. Most processing is accomplished by executing some standard set of queries based in part on past in-depth analysis. This may mean that no "unexpected" findings are likely to be discovered since the search process is static, unaffected by changes in the database or the user's perspective. In addition, traditional data analysis is characterized by a one-way flow in which data is extracted from a database using a query language, stored in intermediate files, and subsequently analyzed using a particular statistical package. Knowledge discovery tools (KDTs) integrate database access methods and exploratory data analysis, often relying on domain knowledge for direction.[1] That is, there is a two-way interaction in which the results of one analytic step may be used to guide subsequent steps, possibly modifying the domain knowledge itself. Discovery tools are intended to leverage an analyst's efforts by supporting an iterative discovery process.

Frawley et al. describe a generic discovery tool architecture [FPSM91]. In this architecture, the role of domain knowledge, as well as user interaction, in focusing the search process is clearly shown. The prototype AX knowledge discovery tool can be usefully viewed from this perspective. Domain knowledge, describing meaningful abstractions or database views, is explicitly maintained by users. These concepts are then linked together for use in a search process that lends some automation to the discovery tool. Four important aspects of knowledge discovery tools derived from the generic architecture include: the *language* used to represent knowledge, the ability to describe the *accuracy* or probablistic nature of knowledge, measures of the "interestingness" of knowledge, and concerns for *efficiency* [FPSM91].

We represent knowledge or discovered *regularities* using the language of logic-based *rules*. Predicates are the primitive *concepts* or database views which are associated in a rule structure. For example, a rule from the domain of bank loan applications might indicate that "living in a bad neighborhood is associated with loan rejection, despite a lot of savings."

$$bad\_neighborhood(Person) \wedge large\_savings(Person) \rightarrow loan\_rejected(Person) \quad (1)$$

[1]Another term for discovery software that is finding favor is *siftware*.

The following section describes our rule representation and several measures of rule quality, which address the issue of accuracy. In Section 3, the representation and organization of concepts or domain knowledge is discussed. The discovery process based on heuristic search is presented in Section 4, including issues relating to "interestingness" and efficiency.

2 Knowledge as Rules

The knowledge or patterns we wish to discover are probabilistic relationships between two or more concepts. In this Section, we develop a more formal representation for expressing such relationships or regularities. We represent such patterns or regularities as rules, adapting the form proposed by Dhar and Tuzhilin [DT93]. That is, knowledge is expressed by a rule of the following form.

$$p_1 \wedge \ldots \wedge p_n \rightarrow q \; [RC, RS, RP] \tag{2}$$

Where p_i, $i = 1, \ldots, n$ are database relations or user-defined predicates (or their negations), and q is a relation, user-defined predicate, or relational operator $(=, <, \leq, \geq, >)$. Once a rule is proposed, how should we assess the quality of the regularity? That is, what does it mean to evaluate a rule? The measures of rule quality we use include *rule certainty* (RC), *rule scope* (RS), and *rule power* (RP), which can be combined to form a single measure of rule quality.

Rule certainty, RC, is defined as the conditional probability of the head of a rule being true given that the body of the rule is true. That is, the certainty of rule $p \rightarrow q$ is $P(q \mid p)$.

$$RC = P(q \mid p) = \frac{P(p \wedge q)}{P(p)} = \frac{\mid p \wedge q \mid}{\mid p \mid} \tag{3}$$

Where $\mid p \wedge q \mid$ is the number of tuples satisfying conditions p and q and $\mid p \mid$ is the number of tuples satisfying condition p.[2]

While the measure of certainty discussed above is important, it may be misleading if the number of tuples "covered" by the rule is small. How useful is a nearly certain rule if it only applies to a single tuple? In order to address this concern, we use a measure of rule scope, RS, to indicate how widely the rule is applicable. Scope is defined as $P(p)$, the probability of a tuple meeting the premise p.

$$RS = P(p) = \frac{\mid p \mid}{N} \tag{4}$$

Where N is the total number of tuples. A related measure of "support" or frequency has been useful in other systems, for instance see [AIS93] [MM93]. We can use RC and RS to calculate *rule frequency*, RF.

$$RF = RC \cdot RS = \frac{\mid p \wedge q \mid}{\mid p \mid} \cdot \frac{\mid p \mid}{N} = \frac{\mid p \wedge q \mid}{N} \tag{5}$$

[2]The symbol p is used here as an abbreviation for the antecedent $p_1 \wedge \ldots \wedge p_n$.

Rule power, RP, complements rule certainty, having the same numerator, but the number of tuples satisfying the head of the rule as the denominator.

$$RP = \frac{P(p \wedge q)}{P(q)} = \frac{\mid p \wedge q \mid}{\mid q \mid} \tag{6}$$

Where $\mid p \wedge q \mid$ is the number of tuples satisfying conditions p and q and $\mid q \mid$ is the number of tuples satisfying condition q. Intuitively, this measure can be thought of as the proportion of the consequent, q, explained by the rule.

The above measures are used to rank discovered rules, both as single measures and in combination as rule quality. Assessing rule quality is important for effectively browsing large collections of rules, as well as in the heuristic search process described in Section 4

3 Domain Knowledge

Predicates are the primitive fragments of "knowledge" from which more complex rules are constructed. In this section, we will formalize the notion of a predicate as a tool for expressing concepts meaningful to a user. How do we define predicates that implement views of the underlying data? Once we define a set of predicates, how can the set be organized to support the knowledge discovery process?

Predicates such as "bad neighborhood" are meaningful abstractions on the underlying data. We call these abstractions *concepts*, a term used in the machine learning literature. The "knowledge" we are attempting to uncover is expressed as rules relating two or more concepts. Therefore, we need a representation scheme for defining concepts, which are similar to relational *views* [Ull88]. Dhar and Tuzhilin [DT93] propose *user-defined predicates* as a means of representing concepts. User-defined predicates are disjunctions of conjunctive clauses, $\bigvee_{i=1}^{m}(\bigwedge_{j=1}^{n} p_{ij})$, where each term, p_{ij}, is either a database relation, relational operator, or another user-defined predicate. For example, the concept $rich(Person)$ may be defined in terms of the underlying $CREDIT$ relation, a previously defined $large\_estate$ concept, and restrictions on savings and salary.

$$\begin{aligned} rich(Person) \leftarrow\ & CREDIT(Person, \ldots, Loan) \wedge \\ & large\_estate(Person) \vee \\ & [Savings > 75 \wedge Salary > 100] \end{aligned} \tag{7}$$

Vocabularies of user-specified concepts have two levels of organization. A single predicate is defined as a disjunction of conjunctive clauses and can be viewed as an AND/OR-tree. This structure offers a limited form of new concept "discovery" since the intermediate AND-nodes can be named by machine and used as concepts themselves. The vocabulary of user-defined predicates can be further organized as higher-level structures—called *concept networks*. A fragment of a vocabulary is defined below to make the discussion more concrete.

$$middle\_class(Person) \leftarrow CREDIT(Person, \ldots, Loan) \wedge$$

$$[Savings \leq 75 \wedge Savings > 25] \quad (8)$$

$$poor(Person) \leftarrow CREDIT(Person, \ldots, Loan) \wedge Savings \leq 25 \quad (9)$$

$$\begin{aligned}
upper\_middle\_class(Person) &\leftarrow middle\_class(Person) \wedge Savings > 60 && (10)\\
lower\_middle\_class(Person) &\leftarrow middle\_class(Person) \wedge Savings < 40 && (11)\\
other\_middle\_class(Person) &\leftarrow middle\_class(Person) \wedge \\
&\quad \neg upper\_middle\_class(Person) \wedge \\
&\quad \neg lower\_middle\_class(Person) && (12)
\end{aligned}$$

$$\begin{aligned}
large\_estate(Person) &\leftarrow CREDIT(Person, \ldots, Loan) \wedge \\
&\quad Property > 1000 && (13)\\
small\_estate(Person) &\leftarrow CREDIT(Person, \ldots, Loan) \wedge \\
&\quad Property < 100 && (14)\\
other\_estate(Person) &\leftarrow CREDIT(Person, \ldots, Loan) \wedge \\
&\quad [\neg large\_estate(Person) \wedge \\
&\quad \neg small\_estate(Person)] && (15)
\end{aligned}$$

3.1 Concept Networks

Dhar and Tuzhilin [DT93] propose a classification hierarchy for user-defined predicates, based on a partial order derived from logical implication. We can incorporate classification hierarchies in a more complex structure—a concept network. The nodes in a concept network are user-defined predicates, which represent meaningful user-level concepts. These concepts are interconnected using three types of links.

1. *Classification links* form a classification hierarchy, they indicate that a concept may be unambiguously classified by a *mutually exclusive* set of child concepts, and are represented as solid lines.[3] A complete taxonomy can be ensured by assuming an implicit default category defined as the negation of the original child concepts.

2. *Composition links* indicate that a given concept is composed, at least in part, of some set of underlying concepts (represented as dashed lines). A concept is "composed of" other user-defined concepts if they appear on the "right hand side" of the definition.

3. *Association links* indicate that a particular concept is simply associated with other concepts, and are represented as dotted lines. This type of link allows users to make explicit associations that seem appropriate.

[3]Classification schemes must be exclusive, but it should be noted that a given tuple may classified using many different schemes. If the classifications are not mutually exclusive, there may be problems of "double counting" tuples.

Concept networks are general graphs rather than strict hierarchies or tree structures. Therefore, we can explore a region of related views around a given concept by visiting adjacent or "nearby" concepts in multiple hierarchies. We will use these structures in our attempts to introduce the capability for autonomous exploration into our knowledge discovery systems, an aspect which is discussed in Section 4.

Figure 1 represents a simple network of loan-related concepts, some of which were defined were defined above. The concept *middle_class* can be classified (solid lines) as *upper_middle_class*, *lower_middle_class*, or an implicit concept *other_middle_class*. The concept *rich* is composed (dashed line) in part by *estate_size*. Lastly, *wealth* is connected to the related *neighborhood* concept by a user-specified association link (dotted line).

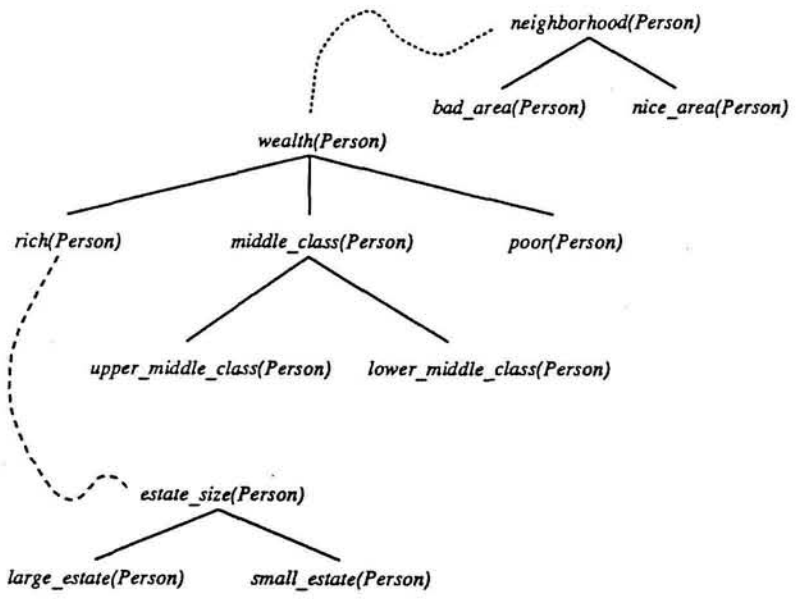

Figure 1: A Network of Loan-related Concepts

4 Discovering Regularities

In the previous two sections, we have considered how to define predicates and their combinations in the form of rules. The heuristic search process is responsible for exploring potential relationships among user-defined concepts. That is, finding the rules that describe these relationships in an automated fashion. However, the space of possible rules can be immense—depending on the type of rules sought and the complexity of the underlying database. Therefore, we approach the problem of transforming concepts into discovered knowledge in a stepwise fashion. Three stages can be identified in this process: sets of concepts are formed (by the user and/or search process), rules are generated from the concept sets, and each rule is evaluated with respect to the underlying database.

The search process begins with the selection of a group of "interesting" concepts—a *concept set*. We can assume that the user makes the initial selection or a collection of "seed" concept sets are generated automatically, say all two-concept sets. The initial concept sets serve as the starting points for the exploration of the search space of concept sets, which is developed through the interplay of heuristic techniques and user intervention. Each node in the search space—a concept set—may give rise to a large number of rules. Section 4.1 presents some methods for generating rules from concept sets.

The classification, composition, and association links in the user-defined concept networks give rise to a group of *concept set operators.* Classification links define both *generalization* and *specialization* operators, depending on the traversal direction. *Decomposition* and *composition* operators are derived from composition links. An association edge is typed and currently supports both *substitute* and *compliment* relationships. A substitute concept calls for replacement, while a compliment concept is added to the set. Lastly, we can include concept *deletion* and concept *addition* as operators. These eight operators can be used to develop the search space, with their application guided by heuristics and user intervention.[4] The search tree fragment in Figure 2 is based, in part, on the previously presented concept networks and the application of the above operators (indicated on each edge).

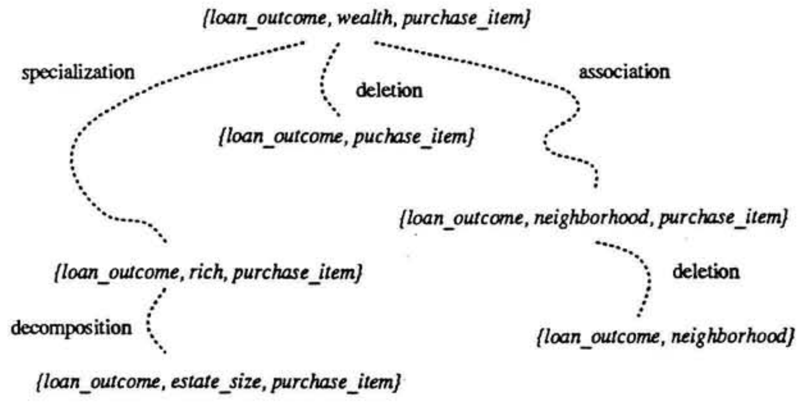
{loan_outcome, wealth, purchase_item}
specialization
deletion
association
{loan_outcome, puchase_item}
{loan_outcome, neighborhood, purchase_item}
{loan_outcome, rich, purchase_item}
deletion
decomposition
{loan_outcome, neighborhood}
{loan_outcome, estate_size, purchase_item}

Figure 2: An Example Search Tree

The heuristic search process entails selecting the best concept set which still offers opportunities for the application of concept set operators. Concept sets are ranked by the quality of the associated rules. The operator application strategy favors less expensive operations, such as deletion and generalization, that make use of existing summary tables as described in Section 4.1. The decision whether to generalize or specialize a particular concept is guided by the number of domain values, controlled by user-specified thresholds (see [HF94]). Lastly, the addition operator can be used to extend a given concept set.

4.1 Deriving Rules from Abstracts

Once a concept set is selected, either by the user or by application of the operators discussed above, our system must generate and evaluate the candidate rules. Each concept set can give rise to a large number of rules—representing discovered knowledge.

[4]The search space may actually be a cyclic graph since duplicate concept sets may be formed through alternative paths. However, by marking duplicates we can manipulate the search space as a tree.

One mechanism we propose using is based on *abstracts* or summaries of the underlying database with respect to a given concept set (see [DT93] for an in-depth discussion of abstracts). An abstract is derived from the concepts and an *aggregation principle*, which determines how the data will be grouped. Examples of aggregation principles include summation, averaging, maximizing, minimizing, and most commonly counting.

Furthermore, we can employ two types of abstracts. A *complete abstract* which summarizes all the data, and a *sampled abstract* based on some type of statistical sampling—thereby improving performance [KM94].

Rules can be derived from abstracts by using statistical tools, such as cluster analysis, or by non-statistical techniques. Dhar and Tuzhilin propose several methods for deriving rules. One algorithm (we term the *FFA* algorithm) is based on "fixing" all attributes except one, the "free" attribute, and comparing across the values of the free attribute. Using the abstract in Table 1, we can fix *loan_outcome* (say to "approved") and allow *purchase_item* to vary. We then fix *loan_outcome* to "rejected" and repeat the process. The following rules are representative of the derivation process, where the associated "quality" vector includes rule certainty, scope, and power as defined in Section 2.

$$medical(Item) \rightarrow approved(Loan)[0.85, 0.21, 0.26] \tag{16}$$

$$luxury(Item) \rightarrow approved(Loan)[0.72, 0.26, 0.27] \tag{17}$$

$$transportation(Item) \rightarrow rejected(Loan)[0.61, 0.16, 0.30] \tag{18}$$

loan_outcome	*purchase_item*	**count**
approved	home_furnishings	94
rejected	home_furnishings	45
approved	luxury	69
rejected	luxury	27
approved	medical	66
rejected	medical	12
approved	transportation	23
rejected	transportation	36

Table 1: An Example Abstract

4.2 Interestingness

The choice of which concept sets to explore is central to the heuristic search process. Our measures of "interestingness" allow us to tradeoff search quality against brute force effort. Furthermore, we can consider interestingness from two perspectives—the perspective of the user and the richness of the data.

Every user will have a different perspective on the interestingness of discovered knowledge. This will be reflected in the specification of any initial concept sets as well as subsequent guidance. In particular, we allow the user to mark concept sets as "hot spots" that can serve as primary areas of focus, as well as mark individual rules as important examples. These user actions can serve as the basis for some measures of interestingness. For instance, the distance from a focal concept set can be used to attenuate "raw" measures of interest. The user may also wish to indicate individual concepts of particular interest, thereby making certain concept sets more interesting. That is, the user may partially specify the type of rules desired.

The underlying data is the ultimate determinant of the quality of rules that can be derived from a given concept set. The number of rules, as well as their strength, are obvious factors in assessing the merit of a given concept set. For those concept sets yet to be explored, the member concepts can be used to estimate interest. Individual concept merit reflects the current crop of rules in which it serves as a component.

4.2.1 Rule Quality

We can combine the rule certainty, rule frequency, and rule power measures to form a *rule quality* metric, RQ. This measure can be used to rank derived rules and guide the search process. A three-dimensional quality space can be formed using the above measures, (RC, RF, RP). The origin is the lowest possible quality score, corresponding to $\mid p \wedge q \mid = 0$. That is, when no tuples satisfy the rule. The best quality score is obtained when all measures are equal to one, yielding a maximum of $\sqrt{3}$. Rule certainty and power can only equal one simultaneously if the sets described by p and q are identical. This is the very "interesting" case of both the antecedent and consequent corresponding exactly. The rule frequency dimension simply indicates that the more tuples supporting the rule, the better (holding other factors constant). So, RQ is at a maximum if the antecedent and consequent sets coincide and cover the entire database. Therefore, rule quality can be interpreted as a distance in three-dimensional space, defined as follows.

$$RQ = \sqrt{RC^2 + RF^2 + RP^2} \tag{19}$$

This measure is zero when the antecedent and consequent are disjoint, and satisfies two other principles of rule interest proposed by Piatetsky-Shapiro [PS91]. That is, RQ monotonically increases with $\mid p \wedge q \mid$ and monotonically decreases with $\mid p \mid$ or $\mid q \mid$, holding other parameters constant. Rule quality may then be used in conjunction with other interest measures, for example see [PS91].

5 Conclusions

In summary, this paper outlines the heuristic search process being developed in the AX knowledge discovery tool. The search process rests on explicit domain knowledge, represented as user-defined predicates or concepts, organized as concept networks. In addition, the search process is conducted at the level of concept sets, using abstract-driven discovery to evaluate individual rules [DT93]. Three important rule quality measures are used within the system: rule certainty (conditional probability), rule scope, and rule power. In particular, the interaction of rule certainty and power provides an indication of how closely the rule antecedent and consequent correspond. These three measures define a three-dimensional interest space which can be used to rank rules, and thereby guide the search process. An example loan application database served to illustrate the process, and was obtained from the machine learning repository maintained at the University of California at Irvine.

References

[AIS93] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD*, May 1993.

[DT93] Vasant Dhar and Alexander Tuzhilin. Abstract-driven pattern discovery in databases. *IEEE Transactions on Knowledge and Data Engineering*, 5(6), December 1993.

[FPSM91] William J. Frawley, Gregory Piatetsky-Shapiro, and Christopher J. Matheus. Knowledge discovery in databases: An overview. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, *Knowledge Discovery in Databases*, pages 1–27. The AAAI Press, Menlo Park, California, 1991.

[HCC92] Jiawei Han, Yandong Cai, and Nick Cercone. Knowledge discovery in databases: An attribute-oriented approach. In *Proceedings of VLDB-92*, pages 547–559, 1992.

[HF94] J. Han and Y. Fu. Dynamic generation and refinement of concept hierarchies for knowledge discovery in databases. In *Proceedings of the AAAI-94 Workshop on Knowledge Discovery in Databases (KDD-94)*, pages 157–168, 1994.

[KM94] Jyrki Kivinen and Heikki Mannila. The power of sampling in knowledge discovery. In *Proceedings of the ACM Symposium on Principles of Database Systems*, pages 77–85, May 1994.

[MM93] John A. Major and John J. Mangano. Selecting among rules induced from a hurricane database. In *Proceedings of the AAAI-93 Workshop on Knowledge Discovery in Databases*, pages 28–44, 1993.

[PS91] Gregory Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, *Knowledge Discovery in Databases*, pages 1–27. The AAAI Press, Menlo Park, California, 1991.

[Ull88] Jeffrey D. Ullman. *Database and Knowledge-base Systems*, volume 1. Computer Science Press, Rockville, Maryland, 1988.