IDEA GROUP PUBLISHING
IGP
701 E. Chocolate Avenue, Suite 200, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com
ITP4476

Handling Large Databases in Data Mining

M. Mehdi Owrang O.
American University, Dept of Computer Science & IS, Washington DC 20016
owrang@american.edu

ABSTRACT

Current database technology involves processing a large volume of data in order to discover new knowledge. The high volume of data makes discovery process computationally expensive. In addition, real-world databases tend to be incomplete, redundant, and inconsistent that could lead to discovering redundant and inconsistent knowledge. We propose to use domain knowledge to reduce the size of the database being considered for discovery and to optimize the hypothesis (representing the pattern to be discovered) by eliminating implied, unnecessary, and redundant conditions from the hypothesis. The benefits can be greater efficiency and the discovery of more meaningful, non-redundant, non-trivial, and consistent rules.

1. INTRODUCTION

Modern database technology involves processing a large volume of data in databases in order to discover new knowledge. Knowledge discovery in databases (KDD) is defined as the non-trivial extraction of implicit, previously unknown, and potentially useful information from data [1,2,4,5,12,14,16]. While promising, the available discovery schemes and tools are limited in many ways. Some databases are so large that they make the discovery process computationally expensive. Database containing on the order of $N=10^9$ records are becoming increasingly common, for example, in the astronomical sciences. Similarly, the number of fields can easily be on the order of 10^2 or 10^3, for example, in medical diagnostic applications [4]. If we apply discovery algorithms to discover all the correlation between concepts in a real database, we will generally observe the production of a set of results whose size is just too large to be handled in a useful manner. Another major concern in the knowledge discovery is the consistency of the databases. Databases include redundancies that could lead to discovering redundant knowledge.

The vastness of the data and the redundancy that exists in databases force us the use of techniques for optimizing the process for discovering consistent and useful patterns. Existing optimization techniques include using parallel processor architecture, providing some measure of "interestingness of patterns", eliminating irrelevant attributes, data sampling, data segmentation, and data summarization to reduce the size of the databases being considered for discovery as well as to define a bias in searching for interesting patterns. These techniques are overviewed in Section 2.

The human user almost always has some previous concepts or knowledge about the domain represented by the database. This information, known as domain or background knowledge, can be defined as any information that is not explicitly presented in the data [1,4,9,16], including the relationship (or lack of it) that exists among attributes, constraints imposed on data, and redundant data definition. In this paper, we propose a new approach based on domain knowledge to reduce the size of the database being processed for discovery by eliminating the records that are irrelevant to the specific discovery case. In addition, we discuss how to use domain knowledge to optimize the hypothesis as well as the database query used to prove/disprove the hypothesis that represents the interesting knowledge to be discovered.

2. INHERENT PROBLEM IN VERY LARGE DATABASES

Knowledge discovery systems rely on databases to supply the raw data for input, and this raises problems in that databases tend to be dynamic, incomplete, noisy and large. The problems associated with the large volume of data includes [1,4,5,10,12]:

Size of the database: Databases with hundreds of fields and tables, millions of records, and multi-gigabyte size are quite common, and terabyte databases are becoming to appear [1,4,5,14].

High dimensionality: There are often a very large number of records in the database. In addition, there can be a very large number of fields (attributes, variables) so that the dimensionality of the problem is high. A high dimensional database creates problems in terms of increasing the size of the search space for discovering rules in a combinatorially explosive manner [1,4]. In addition, it increases the chances that a data mining algorithm will find spurious patterns that are not valid in general.

Irrelevant attributes: Another issue is the relevance or irrelevance of the attributes involved in the current focus of discovery [1,4]. For example, height and diagnosis may not be causally related if the discovery focus deals with liver disease, but they may be casually related for physiotherapy.

Overabundance of patterns: When search for patterns has a wide scope, a very large number of patterns can be discovered. In knowledge discovery, the concepts of "large databases" and "useful patterns" often interact in a seemingly paradoxical way [1,10]. On one hand, the larger a database, the richer its pattern content and as the database grows, the more patterns it includes. On the other hand, after a point, if we analyze too large a portion of a database, patterns from different data segments begin to dilute each other and the number of useful patterns begins to decrease.

3. APPROACHES TO THE OPTIMIZATION OF DISCOVERY PROCESS

The goals for the optimization of the KDD process can be identified as:

1. To reduce the size of the database being considered for discovery, which leads to faster response.
2. To avoid discovering inconsistent, redundant, and trivial knowledge, thereby minimizing search efforts.
3. To define more efficient data mining algorithms.
4. To employ parallel processing architecture to improve the discovery process.

In the following, we overview the existing approaches to the optimization of the KDD process. The main issue/concern in the optimization of the KDD is the performance versus the accuracy of the KDD process. That is, any of the following optimization technique (or combination) can improve the performance; however, the final discovery results may suffer when the chance of blocking unexpected discovery increases. For example, when

This paper appears in *Challenges of Information Technology Management in the 21st Century,* the proceedings of the Information Resources Management Association International Conference.

we use data sampling, we throw away data not knowing what we discard and the discarded data may indeed contains the unexpected discovery.

3.1 Interestingness of Patterns

Databases contain a variety of patterns, but few of them are of much interest. One approach to solve the problem of searching the large databases is to provide some measure of "interestingness of patterns" and then search only for patterns interesting according to this measure [1,12,13,16]. Piatetsky et al. talk about methods employed in knowledge discovery that has the ability to find patterns according to some measure of "interestingness"[13]. Interestingness refers to the degree of which a discovered pattern is of interest to the user of the system and is driven by factors such as novelty, utility, relevance, and statistical significance [1,13,16]. An automated discovery system requires specific interestingness factors which it can measure, as well as a way of combining these factors into a metric that accurately reflects how domain experts judge key patterns.

3.2 Elimination of Irrelevant Attributes

Eliminating the attributes that do not participate in the discovery can reduce the size of the database [4,16]. Ziarko [16] uses the theory of rough set for the identification and the analysis of data dependencies or cause-effect relationships in databases. He demonstrates how to evaluate the degree of the relationship and identify the most critical factors contributing to the relationship. Identification of the most critical factors allows for the elimination of irrelevant attributes prior to the generation of rules describing the dependency.

Subramanian [15] proposes an irrelevance principle to minimize a formulation by removing all facts that are either logically or computationally irrelevant to the specified discovery case. However, in attempting to discover, facts cannot be removed on the basis of such an irrelevance principle because the concepts to be discovered might precisely involve knowledge initially considered as irrelevant.

3.3 Data Sampling

Limiting the number of fields alone may not sufficiently reduce the size of the data set, in which case a subset of records must be selected [1,5,7,11]. We must use proper sampling techniques (i.e., random sampling) to obtain the data sets to avoid bias in the samples. Obviously a sample should be significantly smaller than the original data set, but if the sample is too small, it contains many spurious regularities, and much additional work is needed in the validation. In [7,8], authors address the question of sufficient sample size. There are several approaches to sampling, including static sampling, dynamic sampling, cluster sampling, etc [1,5,7,8].

When we sample data, we lose information, because we throw away data not knowing what we keep and what we ignore. Sampling will almost always results in a loss of information, in particular with respect to data fields with a large number of non-numeric value [11]. The rules discovered in a sample data can be invalid on the full data set. Statistical techniques, however, can measure the degree of uncertainty. Piatetsky [12] presents a formal statistical analysis for estimating the accuracy of sample-derived rules when applied to a full data set.

3.4 Data Summarization

Summarization techniques can be used to reduce database size [1,4,5,11]. In general, summary tables hold pre-aggregated and pre-joined data. Basically, for any given detailed data, there are numerous ways to summarize it. Each summarization or aggregation can be along one or more dimensions. For the general case, given N items (or columns), there are 2^N-1 possible ways of combining the items.

Just processing the summary tables may not discover accurate knowledge. The problem is that, the summarization of the same data set with two summarization methods may result in the same result, and the summarization of the same data set with two methods may produce different results. The following example shows how "information loss" and "information distortion" can take place through summarization [11]. Consider a retail database where Monday to Friday sales are exceptionally low for some stores, while weekend sales are exceptionally high for others. The summarization of daily sales data to weekly amounts will totally hide the fact that weekdays are "money loser", while weekends are "money makers" for some stores. In other words, key pieces of information are often lost through summarization, and there is no way to recover them by further analysis.

3.5 Data Segmentation

Most of the time it does not make sense to analyze all of a large database because patterns are lost through dilution. To find useful patterns in a large database, we usually have to select a segment (and not a sample) of data that fits a business objective, and then perform data mining [1,4,5]. Looking at all of the data at once often hides the patterns, because factors that apply to distinct business objectives often dilute each other. As we segment, we deliberately focus into a subset of the data (e.g., select one model year for a car, or select one marketing campaign), sharpening the focus of the analysis. For instance, responses to campaigns for a new checking account may have little bearing on responses to campaigns for a new credit card or refinancing a home. By considering more data, we lose accuracy since some of the data will not be relevant to the task we are considering.

The main concern in segmentation is what happens if there are one or two key indicators that are common to all of the campaigns and whether they will be lost if we just analyze the campaigns a few at a time. The answer is no (in most cases), because if a pattern holds strongly enough in the entire database, it will also hold in the segment. For example, if the people with more than five children never respond to campaigns, this fact will also be true in each individual campaign.

3.6 Parallel Processing

Advanced parallel hardware and parallelized data-mining algorithms can be used to handle very large databases [1,2,4,10]. Inherent parallelism exist in the data mining algorithms which provide us some flexibility in choosing a particular parallelism scheme that most suited for a specific parallel machine. In [2], authors identify two major schemes for exploiting parallelism within data mining algorithms as *task parallelism* and *data parallelism*. In the task parallelism approach, the computation is partitioned amongst the processors of a particular machine with each processor computing a distinct part of a learning model before co-ordinating with the other processors to form the global model. In the data parallelism approach, the training set is partitioned amongst the processors with all processors synchronously constructing the same model; each operating on a different portion of the data set. For example, in a classification tree algorithm, each branch of the tree is formed into a task (task parallelism); and for data parallelism, training set is partitioned across the processors, and processors evaluate a node of tree in parallel.

The experimental results in [2] indicates that while the parallelization of certain data mining algorithms shows a consis-

tent performance behavior when applied to different data sets, this is not necessarily true across all algorithms, For example, for induction-based classification algorithms, there appear to be no ideal scheme for their parallelization. The performance of the different parallelization scheme varies greatly with the characteristics of data set to which the algorithm is applied. Another problem is the administrative complexity of a system with any number of parallel processors. In addition, a major issue is how to distribute the data among different processors and how to integrate the results produced by different processors.

3.7 Shortcomings of the Existing Techniques

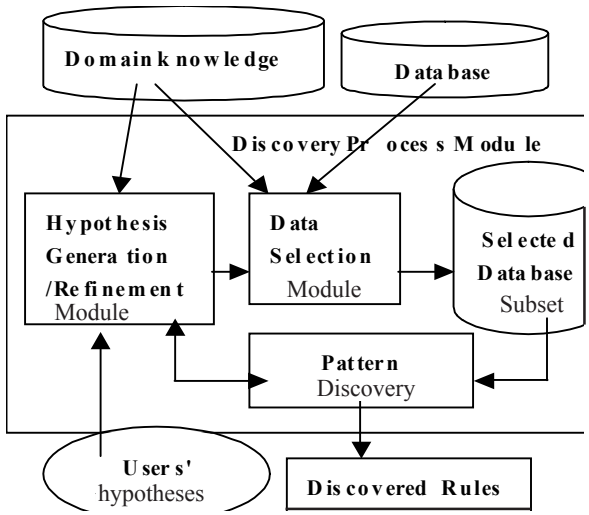
A major problem with some of the above techniques (i.e., sampling, interestingness of patterns) is that they reduce the size of the database by throwing away records/attributes without knowing exactly what they are throwing away. The consequence is the increased chance of missing the discovery of some of the hidden patterns. Another major problem associated with all of the above techniques is the lack of any optimization at the discovery process level.

The heart of the knowledge discovery is the hypothesis that represents the pattern to be discovered. Most of the problems mentioned in Section 2 can be avoided by a well-defined hypothesis; one that does not include any redundant and/or inconsistent conditions relating the attributes in the patterns. In the following section, we discuss our approach of using domain knowledge in reducing the size of the database for discovery cases as well as in defining more consistent, accurate, and efficient hypotheses in order to optimize the knowledge discovery process. In using domain knowledge, we knowingly throw away data, records/attributes as well as the conditions used in the hypotheses, which do not participate in the specific discovery cases, to discover patterns. Figure 1 shows the overall view of the discovery process optimization using domain knowledge.

4. USING DOMAIN KNOWLEDGE FOR OPTIMIZATION OF KNOWLEDGE DISCOVERY PROCESS

Although a database stores a large amount of data, usually only a portion of it is relevant to the discovery task. For example, to find the factors causing "Ovarian Cancer", we can eliminate male patients from discovery consideration since male patients cannot get ovarian cancer (a medical fact, considered as a domain knowledge). Domain or background knowledge can be defined as any information that is not explicitly presented in the database [1,4,5,9,16]. In a medical database, for example, the knowledge "male patients can not be pregnant" or "male patients do not get ovarian cancer" is considered to be domain knowledge since it is not contained in the database directly. Similarly, in a business database, the domain knowledge "customers with high-income are good credit risks" may be useful even though it is not always true.

Figure 1. Overall view of the discovery process optimization using domain knowledge.



Domain knowledge can take different forms. A few examples include: lists of relevant fields on which to focus for the discovery purposes; definition of new fields (e.g., age = current_date - birth_date); lists of useful classes or categories of fields or records (e.g., revenue fields: profits, expenses,...); generalization hierarchies (e.g., A is-a B is-a C); functional or causal dependencies. Formally, domain knowledge can be represented as X Y (meaning X implies Y), where X and Y are simple or conjunctive predicates over some attributes in the database.

4.1 Using Domain Knowledge to Reduce Database Size

Domain knowledge can be used to reduce the size of the database that is being searched for discovery by eliminating data records that are irrelevant to a discovery case. The following example shows how domain knowledge can be used to focus on a proper subset of the database for the discovery case.

Example 1:

Consider a medical database in which we are interested in finding out the factors affecting an individual in developing ovarian cancer. Through medical research we know that only women can develop ovarian cancer. We can use this as domain knowledge to select female patients into data set for KDD and then search for clusters of characteristics that are highly correlated (positively or negatively) with the presence of ovarian cancer in the patient. To formalize the process, assume the set of domain knowledge is represented as:

DK={(cancer_type=ovarian) (sex=female), (cancer_type=ovarian) (age > 20), ... }.

The initial hypothesis (note that the actual hypothesis for discovery may include other attributes of the patients, e.g., race, weight, etc.) can be represented as a rule as follows:

IF Using_Birth_Control_Pills = No AND Using_Fertility_Drugs=Yes AND Family_Member="had ovarian cancer" AND age > … AND …..
THEN cancer_type=ovarian.

The database reduction process can apply the domain knowledge to the initial hypothesis to create a set of constraints. Basically, for each condition (or goal) in the hypothesis, the reduction process searches the set of domain knowledge. If the condition is found to be in the Y (or X) part of a domain knowledge, then the X (or Y) part of the domain knowledge is selected as a constraint. The set of constraints can then be used to create an SQL statement to be executed in order to produce the reduced database. For the above hypothesis, the following SQL statement can be created and executed to produce the reduced database.

Select * Into Reduced-Patient-Relation
FROM Patient-Relation Where sex = 'female' AND age > 20

4.2 Optimization of the Hypothesis in the Knowledge Discovery Process

Knowledge discovery can be perceived as a search problem in which we have to find the correct hypothesis (pattern to be discovered). The basic form of representing a hypothesis is the rule

representation as:

IF $P_1, P_2, P_3, \ldots, P_n$ THEN C, where "," means "and", and P_i is a proposition of the form : *attr* OP *value,* where *attr* is the name of an attribute in the data set, *value* is a possible value for *attr*, and OP { =, -, •, •, <, > } is the operation. C is the consequent of the form *attr* OP *value*.

When there are only a few different hypotheses, it is possible to find and test the correct hypothesis simply by enumeration. In the case of several thousands or even an infinite number of hypotheses, enumeration would not be such a good strategy [1]. Our approach to optimize the hypothesis is to use domain knowledge to eliminate implied, unnecessary, and redundant conditions from the hypothesis. For each pattern or relationship to be discovered, one would [1,10]:

1) Form hypotheses
2) Make some queries to the database
3) View the result and modify the hypotheses if needed
4) Continue this cycle until a pattern merges

Initially, hypotheses can be formed by domain experts (or by the discovery system automatically) and should aim toward one specific concept. To discover consistent and accurate knowledge, we (or the system, if the discovery system automatically generates the hypothesis) need to define the hypothesis accurately and efficiently. The propositions in the hypothesis involve attributes that are normally related to each other, implied by each other, or not related at all. The information regarding the relationship among the attributes (known as domain knowledge) can be used to eliminate unnecessary, implied, and redundant propositions from the hypothesis; thereby, producing less but more accurate and consistent rules. Algorithm 1 shows the basic steps in the hypothesis optimization process.

Algorithm 1: Hypothesis Optimization Algorithm:
Begin
Let DK be the set of all available domain knowledge (defined or derived);
{ $(C_i,C_j) \mid C_i \quad C_j$, i • j };
Let C be the set of all conditions in the premise of the hypothesis;
Let R be the Conclusion of the hypothesis;
For every (C_i,C_j) in DK do
If C_i C and C_j C Then C = C - C_j;
/* eliminate implied, uninteresting, redundant, and trivial conditions
If C_j = R Then C = C ≈ C_i ;
/* expand the hypothesis to include more accurate conditions
End.

4.3 Optimized Query Used to Prove Hypothesis

To discover patterns, a discovery system forms the hypotheses and makes queries to the database and views the result and modifies the hypotheses if needed. The queries can be posed in SQL, a standard query language for many relational databases [3]. Domain knowledge can be used to optimize a query used to prove a hypothesis. For example, consider the following data relations in a database:

employee(E#,Ename,title,experience,seniority)
money(title,seniority,salary,responsibilities)

Assume the knowledge to be discovered is " What are the criteria for an employee to earn more than $50000". An expert may suggest that experience and seniority are the two criteria contributing to having a salary more than $50000. The hypothesis may be represented as the following rule:

IF has experience AND has seniority THEN earn more than 50000 .

To prove (or disprove) the hypothesis, a discovery system may execute the following SQL statement:

Select experience,seniority From employee E,money M
Where salary >= 50000 AND E.title=M.title
AND E.seniority=M.seniority.

Now, assume that we have the following domain knowledge:

Only level-1 and level-2 managers have a salary more than 50000, represented as
(title = level-1) Y (salary >= 50000)
(title = level-2) Y (salary >= 50000)

We can use this domain knowledge to minimize our search by eliminating the unnecessary join operation. Basically, for each condition in the hypothesis, the query optimization scheme searches the set of domain knowledge. If the condition is found to be in the Y part of domain knowledge, then the X part of the domain knowledge will replace the condition and the unnecessary join operation will be removed from the query. The optimized SQL statement for the above example would be:

Select experience, seniority From employee
Where title=level-1 or title=level-2.

4.4 Evaluation of Using Domain Knowledge

4.4.1 Experimental Results

We have done several experiments on the following CAR relation using the IDIS knowledge discovery tool [6].

CAR *(Symboling, Losses, Make, Fuel-Type, Aspiration, Doors, Body, Drive, Engine-Loc, Wheel-Base, Length, Width, Height, Weight, Engine-Type, Cylinders, Engine-Size, Fuel-Sys, Bore, Stroke, Compress, Horse-Power, Peak-RPM, City-MPG, High-MPG, Price)*

We were interested to discover the relationship between the high way mileage and the rest of the attributes. In this experiment, we have eliminated some of the attributes (i.e., Price, Doors), from consideration in the discovery process, based on the available domain knowledge. Some of the domain knowledge was:

- The smaller the Engine-Size, the better the High-MPG
- The lighter the car, the better the High-MPG
- Price of the car is not related to High-MPG
- Engine-Size = Bore * Stroke * Cylinders .

When domain knowledge was used, the discovery process was very fast (it took 3 hours, instead of 2 days without using domain knowledge). Also, the generated rules were fewer but more interesting and non-trivial.

In other experiments, using the national high way accident data (http://www-fars.nhtsa.dot.gov/fars/fars.cfm), the discovered rules were less trivial/redundant using domain knowledge and were generated much faster compared to discovery without using domain knowledge.

4.4.2 Avoid Blocking Unexpected Discovery

Too much reliance on domain knowledge may unduly constraint the knowledge discovery and may block unexpected discovery by leaving portions of the database unexplored. One possible scheme to improve the effective use of domain knowledge in knowledge discovery and to avoid blocking the unexpected discovery is to assign a confidence factor to each domain knowledge and use it only if the confidence factor is greater than a specified threshold. The assignment of a confidence factor to domain knowl-

edge depends on how close the domain knowledge is to the established facts.

5. CONCLUSION AND FUTURE DIRECTION

We have discussed the benefits of using domain knowledge to constrain the search when discovering knowledge from databases. Domain knowledge can be used to reduce the search by reducing the size of the database as well as to optimize the hypotheses by eliminating unnecessary conditions from the hypotheses. In the future, preprocessing must be performed to extract and group the task relevant data from a database before generalization. The preprocessing can be viewed as a relational query which takes a discovery request as a retrieval command to search for the necessary sets of data from the database and group them according to the discovery task. Future discovery tools should be able to look at the nature of data and available domain knowledge in order to produce automatically the retrieval command to search for the relevant data to be processed by the discovery tool.

REFERENCES

1. Adriaans, Pieter and Dolf Zantinge, Data Mining, Addison-Wesley, 1996.
2. Chattratichat, Jaturon, John Darlington, Moustafa Ghahem,"Large Scale Data Mining: Challenges and Responses", Proc. of the third International Conf. on Knowledge Discovery and Data Mining, PP. 143- 46, 1997.
3. Date, C.J., An Introduction to Database Systems, Vol. 1, 5th Edition,Addison-Wesley, Reading, Mass., 1990
4. Fayyad, Usama, "Data Mining and Knowledge Discovery: Making Sense out of Data ", IEEE Expert, Vol. 11, PP. 20-25, 1996.
5. Groth, Robert, *Data Mining: A Hands-On Approach for Business Professionals*, Prentice Hall, 1998.
6. IDIS: the Information Discovery System, User's Manual, IntelligenceWare,Los Angeles,1994.
7. John, George H. and Pat Langley,"Static Vesus Dynamic Sampling for Data Mining", Prec. of the 2nd Inter. Conference on Knowledge Discovery and Data Mining, PP. 367-370, August 1996.
8. Kivinen, Jyrki and Heikki Mannila,"The power of sampling in knowledge discovery", Proc. of the 1994 ACM SIGACT-SIGMOD- SIGACT Symp. On Principles of Database Theory (PODS'94), PP. 77-85, 1994.
9. Owrang O., M. Mehdi,@The Role of Domain Knowledge in Knowledge Discovery in Databases@, Microcomputers Applications, Vol.16, No.1, PP. 11-18, 1997.
10. Parsaye, Kamran,"Large Scale Data Mining in Parallel", DBMS Magazine, March 1995.
11. Parsaye, Kamran,"Small Data, Small Knowledge-The Pitfalls of Sampling and Summarization ", Information Discovery Inc.,http:\\www.datamining.com/datamine/ds-start1.htm.
12. Piatetsky-Shapiro, Gregory,@Discovery, Analysis, and Presentation of Strong Rules@, Knowledge Discovery in Databases, AAAI Press/MIT Press, PP. 229-247, 1991.
13. Piatetsky-Shapiro, G. Matheus,"The interestingness of deviations", Proc. Of the AAAI-94 Workshop on KDD, PP. 25-36, 1994.
14. Simoudis,Evangelos,"Reality Check for Data Mining", IEEE Expert Vol.1,PP. 26-33,1996.
15. Subramanian, D., "A theory of justified reformulation, Change of Representation and Inductive Bias", PP. 147-167, Kluwer Academic Publishing, Boston, 1990.
16. Ziarko, Wojciech, "The Discovery, Analysis, and Presentation of Data Dependencies in Databases", knowledge Discovery in Databases, AAAI/MIT Press, PP. 195-209, 1991.