Distributed Data Mining: Scaling up and beyond

Foster Provost
New York University
New York, NY 10012
provost@acm.org

September 3, 1999

Abstract

In this chapter I begin by discussing Distributed Data Mining (DDM) for scaling up, beginning by asking what scaling up means, questioning whether it is necessary, and then presenting a brief survey of what has been done to date. I then provide motivation beyond scaling up, arguing that DDM is a more natural way to view data mining generally. DDM eliminates many difficulties encountered when coalescing already-distributed data for monolithic data mining, such as those associated with heterogeneity of data and with privacy restrictions. By viewing data mining as inherently distributed, important open research issues come into focus, issues that currently are obscured by the lack of explicit treatment of the process of producing monolithic data sets. I close with a discussion of the necessity of DDM for an efficient process of knowledge discovery.

0.1 Introduction

Until recently, research on distributed data mining (DDM) has been motivated primarily by the desire to mine very large databases. Questioning how often it is necessary to mine huge databases, as opposed to mining a sample of the data, fuels an interesting debate—to which DDM researchers should pay heed. Nevertheless, it is clear that faster data mining sometimes is necessary. Its easy decomposability makes data mining an ideal candidate for parallel processing, and several lessons emerge from existing work.

Although scaling up is an important issue, it also is important for DDM research to look beyond scaling up, where many important research problems lie, relatively unaddressed. Thus, to counterbalance the existing body of work using DDM for scaling up, I argue that DDM is generally preferable to monolithic data mining. Databases *are* distributed. They are heterogeneous. They have privacy restrictions. There often are myriad databases, electronic documents, websites, etc., that might contain information relevant to a particular data mining problem. Building a monolithic database, in order to perform non-distributed data mining, may be infeasible or simply impossible. Hopefully this argument will give additional impetus to research in potentially fruitful areas: mining multitable databases, mining the web, mining with background knowledge, etc.

Viewing *knowledge discovery* as an inherently distributed activity makes existing data mining technology seem only narrowly applicable. Should not data mining algorithms be able to take advantage of all the data, information, and knowledge that is available only a mouse click away? Even a little of it? Currently, researchers are designing meta-data that will facilitate machine access to the vast universe of digital information. We should prepare for automated knowledge discovery.

0.1.1 Lessons from scaling up with DDM

The most common motivation for research on distributed data mining is the need to scale up to massive data sets. Because the run-time complexity of data mining typically is linear or worse in the total number of instances, massive data sets can be prohibitively expensive to mine unless attention is paid to scaling up. Besides simply having a massive data set, other motivations for fast data mining include: interactive induction (Buntine 1991), in which an inductive program and a human analyst interact in real time; mining multiple models and combining their predictions (Dietterich 1997); and wrapper approaches, which for a particular problem and algorithm iteratively search for feature subsets or good parameter settings (Kohavi and Sommerfield 1995; Kohavi 1996; Provost and Buchanan 1995; Provost 1992). In addition to making many runs of a data mining program, experimenting with many machine-learning biases requires a large data set to avoid overfitting due to bias selection (DesJardins and Gordon 1995). Furthermore, in a wrapper approach, each evaluation may involve multiple runs to produce performance

statistics (e.g., with cross-validation).

Venkat Kolluri and I recently surveyed the state of the art of scaling up decision-tree/rule induction (Provost and Kolluri 1999). Four main ideas that emerge from existing work on distributed data mining are as follows.

- Distributing the search space can be problematic. It does not address directly the problem of massive data, and load balancing is difficult and costly. Shared-memory systems can alleviate the problems, because massive data transfers can be avoided.
- Operating on distributed instances can be effective and very efficient, if centralized control is possible. The gathering of the statistics needed to evaluate a particular pattern (e.g., a rule) can be parallelized all the way down to the individual instances, providing tremendous speedups.
- Using a distributed database management system (DBMS) to control the process is not completely straightforward. There are several ways of implementing data mining routines within a DBMS, each with its own tradeoff. Until flexible and effective DBMS mining routines are commonly available, mining within a DBMS will require specialized programming. Also, it has been observed that even with a fast database machine, mining database-resident data directly is often considerably slower than flat-file mining (Musick 1998).
- Cooperation among distributed processes allows effective mining even without centralized control. Two main techniques for cooperation have been particularly effective. Processors can operate independently on subsets of the data, and then combine their models. A processor also can share potential knowledge as it is discovered, in order to get the opinions of the other processors (e.g., their statistical assessments).

In section 0.2 I will discuss and provide references to some instances of work based on these ideas.

0.1.2 Why scale up to massive data sets?

Databases of customer, operations, scientific, and other sorts of data continue to grow rapidly (Fayyad, Piatetsky-Shapiro, and Smyth 1996). Both manual data mining and the direct application of today's mining techniques can be problematic when data sets exceed 100 megabytes (Provost and Kolluri 1997; Huber 1997). Huber observes that "somewhere around data sizes of 100 megabytes or so, qualitatively new, very serious scaling problems begin to arise, both on the human and on the algorithmic side" (Huber, 1997, p. 306).

Scaling up is desirable, because increasing the size of the training set often increases the accuracy of induced classification models (Catlett 1991). In many cases, the degradation in accuracy when mining smaller samples stems from overfitting due to the need to allow the program to find *small disjuncts* (Holte, Acker, and Porter 1989), elements of a class description that cover few data items. In some domains, small disjuncts account for a large portion of the class description (Danyluk and Provost 1993). Overfitting from small data sets also may be due to the existence of a large number of features describing the data. Large feature sets increase the size of the space of models. Searching through and evaluating more candidate models increases the likelihood that, by chance, the program will find a model that fits the data well (Jensen and Cohen 1999), and thereby increases

the need for larger data sets (Haussler 1988). Things get particularly difficult when there are many features *and* there is the need to find small disjuncts. Specifically, because large feature sets lead to large and often sparsely populated model spaces, a program biased to search for models covering special cases can be inundated with small disjuncts from among which it cannot choose.

Some data mining applications are concerned not with predictive modeling, but with the discovery of interesting knowledge from large databases. In such cases, increasing accuracy may not be a primary concern. However, scaling up nonetheless may be an issue. For example, finding small disjuncts often is of interest to scientists and business analysts, because small disjuncts may capture special cases that were unknown previously (the analysts often know the common cases). As with classifier induction, in order not to be swamped with spurious small disjuncts it is essential for a data set to be large enough to contain enough instances of each special case from which to generalize with confidence (Provost and Aronis 1996).

For all its theoretical considerations, the issue of scaling up is inherently pragmatic. For scaling up induction algorithms, the issue is not as much one of speeding up a slow algorithm as one of turning an impracticable algorithm into a practicable one. The crucial issue is seldom "how fast" you can run on a certain problem, but instead "how large" a problem can you feasibly deal with. Both time and space considerations are critical. Time- and space-complexity analyses should address asymptotic complexity as the numbers of instances and features grow. Also important, the absolute size of the main memory with which the computing platform is equipped should be considered. Almost all existing implementations of induction algorithms operate with the training set entirely in main memory; no matter what the computational complexity of the algorithm, if exceeding the main memory limitation leads to virtual memory thrashing, the algorithm will not scale well.

Finally, the goal of the data mining must be considered. Evaluating the effectiveness of a scaling technique becomes complicated if a degradation in the quality of induction is permitted. The vast majority of work on induction algorithms uses classification accuracy as the metric by which different algorithms are compared. In such cases, we are most interested in methods that scale up without a substantial decrease in accuracy. For problems that require mining regularities from the data for purposes other than classification, metrics should be chosen by which effectiveness can be measured (and compared) as the system scales up.

0.1.3 Is subsampling sufficient?

Before moving on discuss existing work more specifically, let me tarry a while on a point that often is not treated thoroughly enough in papers on scaling up to massive data sets.

Data sampling is well accepted by the statistics community, who observe that "a powerful computationally intense procedure operating on a subsample of the data may in fact provide superior accuracy than a less sophisticated one using the entire data base." (Friedman 1997). The question of scalability asks whether the algorithm can process large data sets efficiently, *while building from them the best possible models*. Thus, for anyone wanting to mine a large data set, an important question is: must I process the whole thing? Or will sampling be effective? The answer is: it depends on the data set and the algorithm. Just because a massive database exists does not imply necessarily that you have to mine it all. In practice, as the amount of data grows, the rate of increase in accuracy slows (Frey and Fisher 1999). Whether sampling will be effective depends on

how dramatically the rate of increase slows. Also, different algorithms benefit from additional data to different degrees. For example, Harris-Jones and Haines (1997) observe that the learning curves of classic statistical algorithms tend to level off after relatively few data, in contrast to the learning curves of algorithms considered more typically by machine learning researchers.

Determining how much data to use is difficult, in general, because the smallest sufficient amount depends on factors not known a priori. For example, it depends on the minimum size of the special cases that must be discovered in order to model the phenomenon effectively. However, if one is willing to bias an algorithm (explicitly or implicitly) against finding very small special cases, then recent work on determining sufficient sample sizes for similar data mining problems provides relevant results. For example, Toivonen (1996) and Zaki et al. (1997) discuss the determination of sufficient sample sizes for finding association rules that are no smaller than a predefined size, based on tolerances on the probability of error and the size of the error. A different view of sufficient sample size, that of *sample complexity*, is provided by Valiant's theoretical framework (Valiant 1984; Haussler 1988), which for a given hypothesis space allows the calculation of the number of instances sufficient for inducing with high probability a good approximation to the "true concept," if one exists in the hypothesis space. It should be noted that determining the sample size *sufficient* for induction may say little about how many examples are necessary in practice. Recent work has investigated systems that can determine empirically how many examples are necessary, by starting small and progressively sampling larger subsets until model performance no longer improves (John and Langley 1996; Provost, Jensen, and Oates 1999).

Published work provides differing views of how often real-world classifier learning curves level off before massive data sets are needed. Catlett's work shows that induction from subsets of data decreases accuracy. Despite the advantages of certain sampling strategies, viz., speed-ups and improving the accuracy of the classifier over random sampling in noise-free domains, Catlett concludes that they are not a solution to the general problem of scaling up to very large data sets (Catlett 1991). However, it should be noted that at the time of Catlett's study, "massive" data sets were much smaller than they are today, and processing times much longer.

Oates and Jensen (1997) study inducing decision trees for nineteen data sets, and look specifically at the number of instances necessary before the learning curves reach a plateau. They regard a plateau to have been reached when an accuracy estimate is within a certain tolerance of the maximum (specifically, one percent, in their experiments). Surprisingly, for these nineteen data sets, an accuracy plateau is reached after relatively few training instances. The three data sets for which the most instances were needed were (from the UCI repository (Blake, Keogh, and Merz 1998)): letter-recognition (17,000 instances), led-24 (4500 instances), and census-income (9768 instances).

In another recent study, Harris-Jones and Haines (1997) analyze the relationship between data set size and accuracy for two large business data sets (up to 300,000 instances), by estimating learning curves empirically. They found that while some algorithms level off quite early, in some cases algorithms (decision-tree inducer C4.5 and its successor C5, in particular) continue to show accuracy increases across the entire range of data set sizes. However, the improvements in accuracy at the upper size limit have become quite small, and it is difficult to conclude that they would continue with another order of magnitude increase in data set size. Even if they would, it is important to question whether the benefit of further, diminishing improvements is worth the associated cost (Haines 1998).

Neither these results nor Catlett's provide ample justification for using DDM to scale up. Every data set in both studies

would fit in the main memory of a modern desktop PC, and, given the existence of fast algorithms for mining monolithic data sets (Provost and Kolluri 1999) and the increasing speed of desktop computers, only under very tight time constraints would DDM be necessary.[1] Distributed Data Mining, as a field of study, would benefit from a few prominent examples of the need to scale up beyond reasonable main memory limits.

0.2 Existing methods for scaling up with parallel processing

Questions of necessity aside, data mining is an ideal application for parallel processing. The problem can be decomposed along many dimensions, often to a very fine granularity. The rich decomposability is illustrated by the variety of approaches that have seen success. Tight, fine-grained parallelization has succeeded for massively parallel machines. Coarser, looser coupling has succeeded for collections of stand-alone computers. With some approaches, the distributed processors act independently, only sharing their final results. With other approaches, they cooperate so that individual processors can obtain a global perspective.

0.2.1 Fine-grained parallelization

Fine-grained parallelization can take advantage of two types of decomposability: *search-space parallelization* and *parallel matching*. Data mining can be framed as the search of a very large space of patterns. In search-space parallelization, the space of patterns is decomposed and different processors search different portions in parallel (Cook and Holder 1990), similar to the parallelization of other forms of heuristic search (Kumar and Rao 1987; Rao and Kumar 1987). Load balancing and interprocess communication add additional complexity and overhead. Search-space parallelization should be particularly useful for data mining algorithms that perform massive search, such as the MetaDENDRAL-style, systematic-search rule learners (Buchanan and Mitchell 1978; Clearwater and Provost 1990; Smyth and Goodman 1992; Segal and Etzioni 1994; Webb 1995; Oates, Schmill, and Cohen 1997), and the experiments of Oates, Schmill and Cohen (Oates, Schmill, and Cohen 1997) show that, indeed, speedups of nearly n can be obtained with n processors (they report results with $n = 2$ and $n = 3$). Mining Bayesian networks from data (Cooper and Herskovits 1992) also requires massive amounts of search, and Lam and Segre (Lam and Segre 1997) show that search-space parallelization can allow much larger networks to be found than with traditional, serial approaches.

In general, search-space parallelization does not address the problem of very large data sets, because each processor will have to deal with all the data or will have to subsample. However, recently Zaki et al. (1999) have had success with search-space parallelization of a decision-tree learner; by taking advantage of a shared-memory multiprocessor, they are able to avoid replicating or communicating the entire data set among the processors. Using shared memory also allows the development of effective load-balancing techniques. Galal, Cook, and Holder (1999) recently have had success using search-space parallelism to scale up a scientific discovery system, dealing effectively with load balancing by having a master manage a priority queue of search nodes, serving them to the slaves as needed.

[1] In fact, the experiments of Harris-Jones and Haines were conducted on a dual-processor 133MHz Compaq computer with 256M RAM running Windows NT. The run time for C5 on 283,649 instances was fifty minutes (fifteen minutes for 99,303 instances).

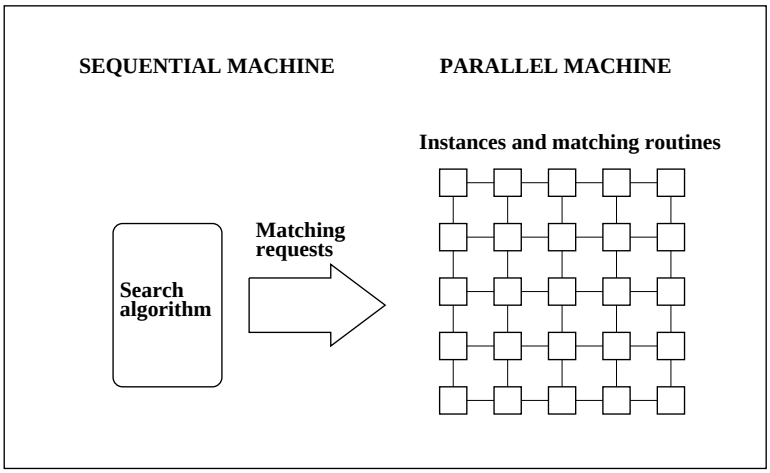

Figure 1: Parallel matching

Massively parallel induction has been more successful when a finer-grained decomposition is used. Parallel matching is based on the observation that search for data mining is different from most other AI searches. In data mining the cost of evaluating a node in the search space is very high, but also is highly decomposable. Nodes (e.g., partial rules or decision tree branches) are hypothesized and each is matched against many instances to gather statistics. With parallel matching, depicted in Figure 1, this compute-intensive matching process is parallelized by migrating the instance set and matching routines to a parallel machine, while the main induction algorithm (the *master*) may run on a sequential front end.

Parallel matching has been used by Lathrop et al. (1990), by Provost and Aronis (1996), and in the parallelization of the SPRINT algorithm (Shafer, Agrawal, and Mehta 1996). The former two efforts use a straightforward parallelization of the matching routines. SPRINT builds data structures called attribute lists, vertical partitions of the data set, used to facilitate efficient construction of decision trees. In the parallel implementation, each processor builds a sublist of each attribute list, and for each decision-tree node sends the master a portion of the statistics needed to determine the best split.

Impressive speedups are reported for parallel matching: less than a minute to mine one million instances on a CM-2 Connection Machine with 8192 bit-slice processors (Provost and Aronis 1996); 400 seconds to mine 1.6 million instances on an IBM SP2 with 16 processors (Shafer, Agrawal, and Mehta 1996).[2] Kufrin (1997) uses parallelization to speed up C4.5's transformation of decision trees to rules (C4.5rules), using parallel matching for two phases of rule-set postprocessing, and dividing up the rule set itself for a third. He also reports impressive speedups (efficiencies averaging more than 0.9 for four induction tasks and up to eight processors).

Zaki (1998) points out that shared-memory multiprocessor (SMP) systems are much more common than massively parallel machines. Motivated by this observation, he presents a parallel matching approach to the design of an SMP version of SPRINT. Instead of distributing the instances, vertical partitions corresponding to SPRINT's attribute lists are distributed and processed in parallel. The attribute lists are divided equally among the processors, which return the matching statistics to the master.

If the data already are resident in a data warehouse with a parallel infrastructure, parallel matching can be done by the

[2] These run times are given for illustration only. No comparison should be inferred.

existing database server (Freitas and Lavington 1996). This approach is fundamentally similar to that shown in Figure 1, except the implementation-specific parallel data representation is replaced by an existing parallel database system. Also, for communication between the front- and back-end, implementation-specific matching requests are replaced with SQL queries. Parallel data mining is treated in more detail by Freitas and Lavington (1997) and by Provost and Kolluri (Provost and Kolluri 1999). Provost and Kolluri also discuss in more detail the use of database systems and SQL queries for scaling up.

0.2.2 Loosely coupled DDM

The previous section addressed fine-grained decompositions, where parallelism is used to speed up existing data mining algorithms so that they feasibly can be run on massive data sets. Alternatively, the data can be partitioned, and mined by a loosely coupled collection of inductive algorithms. Data partitioning techniques separate subsets of instances or subsets of features; viewing the data set as a table, this corresponds to selecting rows versus selecting columns.

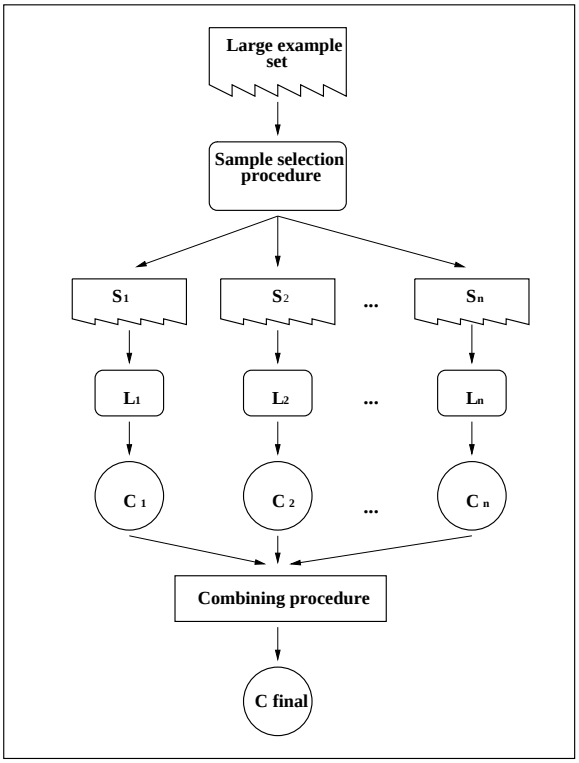

Figure 2: Mining partitioned data

Figure 2 depicts a general model showing the similarities among partitioned-data approaches. Systems using these approaches select subsets $S_1, \ldots, S_n$ of the data based on a *selection procedure*. Data mining algorithms $L_1, \ldots, L_n$ are run on the corresponding subsets, producing concept descriptions $C_1, \ldots, C_n$. Then the concept descriptions are processed by a *combining procedure*, which either selects from among $C_1, \ldots, C_n$ or combines them to produce a final concept description—the

distributed data mining result. Loosely coupled systems differ in the particular procedures used for selection and combining. They also differ in the amount and style of interaction among the mining algorithms.

More precisely, Figure 2 shows a model of *independent multi-sample mining*, because no interaction is depicted between the n mining runs; the C_i are formed independently, and then combined. Fayyad et al. (1993) use a sequential version of independent multi-sample approach in which the L_i are decision-tree inducers; the C_i are rule sets extracted from the decision trees, and the combination procedure is a greedy covering algorithm.

Independent multi-sample mining can be distributed easily. Power is obtained through the combining procedure, which can take place as a sequential post-process, or can be parallelized (as by Kufrin (1997)). Sikora and Shaw (1996) mine multiple rule sets in this manner; their combining procedure is a genetic algorithm. Hall et al. (1998) discuss this approach for building decision trees. Similar to a distributed version of the approach of Fayyad et al., their system builds trees independently from partitioned data, and the trees are converted to rules. The rule sets are merged following the method described by Williams (1990), which resolves conflicts among similar rules. Shasha (1997) and his research group have implemented PC4.5, a parallel version of C4.5 (Quinlan 1993), which uses a simpler instantiation of the framework of Figure 2. Specifically, each C_i is a decision tree built from a different subset of instances. The combining procedure evaluates each C_i on a subset of instances (disjoint from S_i), and chooses the one with the best accuracy as the final concept description.

Chan and Stolfo (1993, 1997) take an independent multi-sample approach in which the L_i can be different induction algorithms, as well as separate instantiations of the same algorithm. Notably, their method forms C_f as a hybrid of the C_i. Instead of constructing C_f by combining selected pieces of the C_i, their approach combines the C_i whole, and makes predictions using a multiple-model, or *ensemble*, approach (Ali and Pazzani 1996). Domingos (1996) also experiments with ensemble combining, finding it superior to taking a simple union of the C_i. The independent multi-sample approach also has been studied from the perspective of learning theory (Kearns and Seung 1995; Yamanishi 1997).

A potential problem with creating a multiple-model hybrid is the resulting loss of comprehensibility. Prodromidis and Stolfo (1998) study several methods for evaluating, composing and pruning hybrid classifiers that reduce size while preserving or even improving predictive performance. A quite different approach to creating comprehensible classifiers from ensembles is taken by Craven (1996), by Domingos (1997), and by Guo and Sutiwaraphun (1998). These authors take advantage of the ability to use machine-learning algorithms to induce understandable models of complex classification systems (Craven 1996) (cf. (Danyluk and Provost 1993)). Specifically, they use the predictions of the ensemble as training labels, and induce from them a decision tree that models the hybrid's performance (with comparable accuracy). The resultant single tree is more understandable than the multiple-model hybrid.

Kargupta et al. (1998, 2000, 1999) consider distributed processing of vertically partitioned data (each processor has a subset of the features, rather than of the instances). Key to the approach is their use of an orthonormal basis-function representation; the coefficients for most of the basis functions can be determined from the partitions independently. If there are critical dependencies among variables that do not reside on the same processor, some communication is required. However, for many data sets the independently learned basis functions may comprise the majority of the function to be learned, and the communication needs will be small. To instantiate their method, they describe how to use various basis functions to build linear discriminant functions, linear regression models, and decision trees. This work relates to the suggestive prior work of

Provost and Buchanan, who show that if the description language is modular, such that useful modules (in their case, rules) can be learned from many different feature subsets, then accurate class descriptions can be built without ever processing a single, suitable subset of features (Provost and Buchanan 1995).

Different from the fine-grained parallel approaches described in section 0.2.1, accuracy may be degraded with these partitioned-data techniques, as compared to running a single induction algorithm with all the data (if that were feasible).

0.2.3 Cooperative DDM

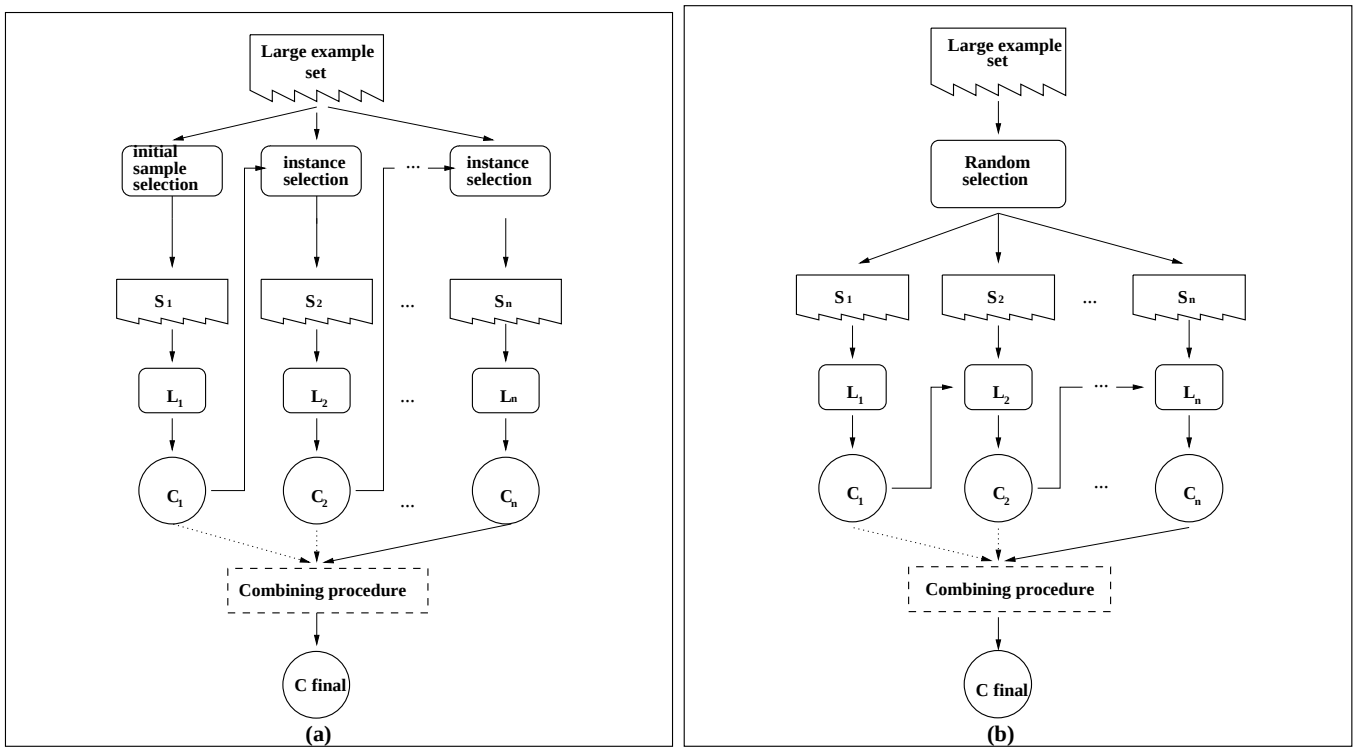

Figure 3: Sequential multi-sample mining: (a) model-guided instance selection, and (b) incremental batch learning

When multiple samples are being processed sequentially, it is possible to take advantage of knowledge mined in one iteration to guide mining in the next. Examples of sequential multi-sample mining techniques include windowing (Quinlan 1983; Fürnkranz 1998), incremental batch learning (Clearwater, Cheng, Hirsh, and Buchanan 1989; Provost and Buchanan 1995; Domingos 1996; Wu and Lo 1998), and some wrapper-based methods for feature selection (Kohavi and John 1997; Kohavi 1996; Provost and Buchanan 1995; Provost 1992), all of which are described in more detail by Provost and Kolluri (Provost and Kolluri 1999). Figure 3 shows two approaches to *sequential multi-sample mining.* In *model-guided instance selection*, shown in Figure 3(a), class description C_i is used in the selection of S_{i+1}. In *incremental batch learning*, shown in Figure 3(b), class description C_i is taken as input to the data mining program and is used in building C_{i+1}.

Concurrency precludes the straightforward parallelization of partitioned-data approaches for which the results from one stage are required as input to the next. Therefore, partitioned-DDM must take a slightly different tack. Rather than assume that results are available before a stage begins, distributed algorithms can cooperate by sharing results as they become available (as depicted in Figure 4). Since many mining algorithms operate naturally as anytime algorithms, producing some results very quickly and then more as time progresses, early in the DDM process there likely will be results that can act similarly to those passed from stage to stage in the sequential mode.

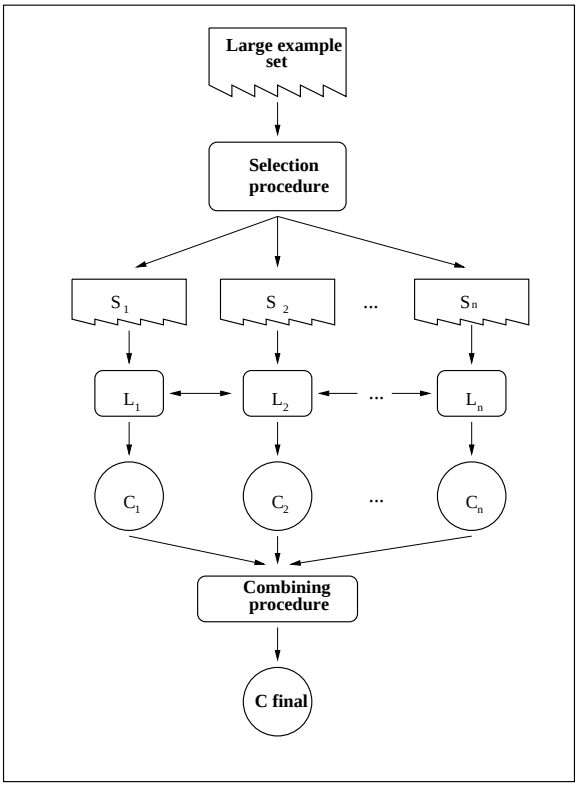

Figure 4: Cooperating DDM

For distributed mining of rules and similar patterns, a useful observation is that certain evaluation metrics obey an *invariant-partitioning property* (Provost and Hennessy 1994; Provost and Hennessy 1996). That is, every rule that is acceptable globally (using the entire data set) according to the metric, will be acceptable on at least one partition. This invariant-partitioning property holds, for example, for many variants of support and confidence (Provost and Hennessy 1994; Provost and Hennessy 1996; Cheung, Han, Ng, Fu, and Fu 1996). By taking advantage of this property, a superset of the rules that satisfy the metric will be generated when all partitions are considered independently. The problem of finding the acceptable rules thus is reduced to selecting from this superset only those rules that are acceptable globally.

Provost and Hennessy (1994, 1996) allow individual processes to cooperate, in order to select all and only the globally acceptable rules. The cooperation takes the form of requests (from the other induction processes or from a server with the

entire database) for verification of statistics regarding the best discovered rules. The combining procedure takes the union of the C_i. Interprocess communication is minimal, because cooperation requests are limited to rules that appear acceptable to at least one induction process, and each cooperation request requires simply that the rule be passed to each processor (which uses very few messages, especially if a logical ring network topology exists within the physical network topology). Provost and Henessy observe speedups linear in the number of distributed processors. The system is effective for very large data sets, up to the point where the individual processors run out of core memory and page thrashing begins (for the data considered, with 64M of main memory, thrashing sets in at several hundred thousand records per processor). Other forms of cooperation, such as sharing pruning information, also are presented.

A similar method is employed by Cheung et al. (Cheung, Han, Ng, Fu, and Fu 1996; Cheung, Ng, Fu, and Fu 1996) to learn association rules. Essential to most association rule learning algorithms is finding rules with support greater than a prespecified level. Since support satisfies the invariant-partitioning property, the search can be distributed, as discussed above. Cheung et al. also discuss efficient cooperation by sharing statistics and by sharing pruning information.

A sequential, but easily distributable, version of this approach is the basis for the algorithm Partition (Savasere, Omiecinski, and Navathe 1995), which has been called one of the most efficient association-rule algorithms in terms of database operations (Toivonen 1996). Similarly, for scaling up a scientific discovery system, Galal, Cook and Holder (1999) found the concurrent/cooperative approach to be the best (among the various techniques they studied). They also partition the problem and then share the best discoveries, which are evaluated by all the processors to obtain a global perspective.

0.3 Beyond scaling up to networked, cooperating knowledge services

The generic model described in the previous section included a sample selection procedure that partitioned a large instance set into several subsets. However, in some cases such an explicit procedure is superfluous, because the data are partitioned and distributed naturally. This is true for traditional databases in corporate environments, many types of public data, and what are becoming known as Digital Libraries (Fox, Akscyn, Furuta, and Legsett 1995). Indeed, distributed data *defines* the most easily accessible data repository of all, the Internet.

Coalescing already-distributed data at best is a time-consuming, partially manual task. In a traditional corporate environment, for example, a product database typically is stored separately from a customer database—often in a different department, in a different building, or even in a different city. The recent trend toward "data warehouses" promises to collect all the relevant data in one, huge, monolithic repository. The promise is encouraging for well-scoped, predefined analyses, such as those typical of OLAP. However, it rings hollow for the purposes of data mining. Experienced data-mining practitioners know that which data are relevant only becomes clear as the knowledge-discovery cycle iterates. Exploration continually provides new prospects for augmenting the data, to improve the chances of making a useful discovery the next time around. Thus, crafting the data set for monolithic data mining often is a substantial part of a data miner's effort. Coalescing data sets consumes much more hands-on time than actually running the mining algorithms.

0.3.1 An example: credit-card fraud detection

Consider a simple, concrete example: a company that issues credit cards wants to mine its customer data for patterns that indicate fraudulent activity (so that it can detect fraud better). What data are relevant for this mining task? Interpreting "data" broadly, this question itself is the crux of knowledge discovery and data mining. Once just-the-right data are described by just-the-right attributes in just-the-right representation language, standard statistical or machine-learning algorithms will compress them into patterns that predict fraud.

Why is this particularly relevant to *distributed* data mining? Consider some of the steps a fraud-detection data miner would take to build just-the-right data set. First, historical customer transactions would be obtained from the billing department. These include customer id, transaction id, transaction date, transaction amount, transaction type, transaction location, etc. What they do *not* include is an indication as to whether the transaction had been fraudulent. There is a separate database detailing which transactions had been fraudulent.[3] To craft his data set, our data miner must cross-reference this secondary table in order to label transactions as fraudulent or legitimate. Further complications arise during the cross-referencing. For example, the data miner notices that the fraud database contains transactions that do not appear in his transaction table! Investigation reveals that there is yet another table of "fraudulent transactions that were never billed to the customer," including those transactions occurring between the date fraud was detected (recall that this is historical data) and the date the previous customer bill had been printed. In order to have a complete picture of the fraud, these data must now be incorporated.

Thus the *initial* data set is built. Soon our hero realizes that he wants more than just transaction data. For example, it seems useful to add information on the customers themselves: a customer's address may help to decide whether or not transactions were made locally; a customer's credit history may help to decide whether a particular balance is unusual. Good database design insists that these factors be stored in tables separate from the transaction data, and, in fact, for organizational reasons they are likely to reside in separate databases. Furthermore, the data miner may want to create his own auxiliary databases, for example, customers' favorite transactions, locations, and merchants, or bandits' favorite transactions, locations, and merchants.

In current data mining practice, much of the "art" of data mining is to craft a single, monolithic table from this inherently distributed information. Unfortunately, producing monolithic tables from multiple, real-world, multitable, relational databases is fraught with problems. The flattening-out process can be quite time consuming; substantial storage space is needed, and keeping the monolithic tables around leads to the problems that relational databases are designed to avoid (e.g., update and delete anomalies). Indeed, flattening may create, from otherwise manageable databases, monolithic tables that can no longer fit in main memory. As an example, consider a database with only three tables: a customer table containing one million customers with twenty fields, including address and product preference; a state table containing fifty states with eighty fields of information on each state; and a product field containing ten products with four hundred fields of information on each product. Furthermore, assume that the average size of a field is five bytes. Even in this vastly oversimplified example, flattening out a 100Mbyte database results in a 2.5Gbyte flat file.

Also notice that coalescing heterogeneous databases usually forces choices regarding which data to use and which data to exclude, choices that may restrict the discoveries that later data mining can make. Consider the realization that additional

[3] Or, more precisely, which transactions had been credited to the customer as being fraudulent, only some of which were actually fraudulent.

geographic information may be helpful to our fraud miner. Transaction locations may be very specific, say at the zipcode level. Based only on this fine-grained information, a data mining program may notice few regularities. However, useful high-level regularities might become apparent if transaction locations were grouped into neighborhoods, or cities, or states, or regions, or countries. Should every transaction be annotated not only with the specific location, but also with the neighborhood, city, county, state, region, country, and continent? If not, how should one decide which safely can be ignored without affecting data mining?

0.3.2 A DDM view of the example

Practitioners will agree that shifting perspectives by incorporating new information from additional databases (or other information sources) is one of the common features of the KDD process. It does not occur more frequently at the beginning of the process, only to taper off, and it is not an indication of incompetent design. Rather, it is a defining feature of exploration, the process leading to discovery. A distributed data mining approach would alleviate many of the problems sketched in the previous example. A data mining program should be told, "the data in database X on system Y is related to the current data in such-and-such a way," and the system(s) should be able to take care of the details. Indeed, a proposal for true "knowledge discovery" from databases would be short-sighted if it did not at least suggest that systems be given the opportunity to frame for themselves such relations.

Of course, viewing data mining as distributed demands that the other databases be easily accessible. It also suggests that the other databases be set up for distributed data mining. Whether or not all databases need special data-mining interfaces depends on the DDM design. Consider as a simple example a DDM design wherein, except for the primary system, all data mining is implemented as SQL queries. Such a system only requires that the other databases have an accessible SQL server.

Thus, with a DDM approach, the problems vanish from our example of data mining for fraud detection. The DDM system starts with a database of historical transactions. In order to get class labels, it queries the database of fraud credits. In order to include the additional transactions, it links to the database of unbilled transactions. While mining, it accesses the databases of customer information and geographic information, in order to find relationships or patterns of similarity. If the data miner creates auxiliary databases of bad locations, or bad merchants, or users' favorites, the DDM system simply can be told of their existence, their relevance, and their location. Never does the data miner face the issue of coalescing databases into a single table, or even a single, multitable database. Excluding relevant information is no longer a worry. The data miner may instead be concerned that existing relevant databases may be overlooked (which, of course, also is a problem for standard, monolithic data mining, but usually is buried far beneath the concerns already discussed).

0.3.3 Privacy restrictions may make monolithic mining impossible

Sometimes databases have privacy restrictions. You may think the answer then is simple: these can not be mined. However, not all restrictions are completely exclusive. They vary in scope and sometimes have complicated rules. Herein lies perhaps the most convincing argument against monolithic data mining: building a monolithic data set may be prohibited. For example, an organization may choose not to, or may not be allowed to, provide access to individuals' data. However, organizations owning

data may be interested in participating in a data mining effort, and may be willing to provide answers to queries for aggregate statistics, even if they are unwilling to share the lower-level data.

In fact, although data mining has raised fearful concern regarding loss of privacy (e.g., consider the KDD-98 panel on Privacy and Data Mining), we should argue that, if done right, data mining should *increase* privacy. As mentioned already, data mining programs need not examine individual records; rather, they need aggregate information. Individual information can be protected on a secure server that only answers certain requests from trusted clients. An individual customer still might ask, why take the chance? Why not just disallow data mining altogether? However, data mining also has potential to enhance privacy. Fraud detection is an obvious example, but consider also junk mail (and junk email, and junk phone calls, etc.), an often-cited privacy concern. If target-marketing data mining were to work perfectly, there would be no "junk." Only consumers who would be interested in a product would be targeted. Of course data mining is unlikely to be perfect, but consider the alternative: relatively blind mass marketing.

Distributed data mining is essential to reconcile these opposing privacy concerns, namely, protecting individual data and reducing unwanted privacy intrusions. Let's continue with our concrete example of credit-card fraud detection, which provides a clear illustration of the conflict. Customers prefer to have their transaction data examined as little as possible, and when it becomes necessary, examination by a computer system (with no ulterior motives) is probably universally preferable—a standard billing system going through our records causes no concern. On the other hand, customers tend to prefer that banks take action to verify usage that appears to be fraudulent. Presumably, customers would prefer that a computer examine their transaction histories (looking for indications of fraud, and nothing else), rather than a bank employee.

At Columbia University, Professor Salvatore Stolfo and his students, in collaboration with several large banks, have been studying data mining for fraud detection (Stolfo, Fan, Lee, Prodromidis, and Chan 1997; Stolfo, Prodromidis, Tselepis, Fan, Lee, and Chan 1997). Indeed banks would like to collaborate, effectively pooling their data, to produce more powerful fraud-detection models. However, they are in fact prohibited by law from sharing individual customers' data. Stolfo et al. report that by taking a distributed approach, the data both can be mined effectively and can be kept secure. As described in Section 0.2.2, models of fraud are mined independently by the individual banks, each bank needing to see only its own customers' data. The models of fraud then are shared and combined. To understand why combining models from different banks can lead to more effective fraud detection, consider two banks whose customers are concentrated in different areas, say New York City and Los Angeles. Mining the New York data, one would discover subtle, local fraud patterns that would not be sufficiently concentrated in the Los Angeles data. However, when Los Angeles customers (or their credit-card numbers) travel to New York, their home bank can use these subtle New York patterns for more effective fraud detection.

This example illustrates distributed data mining across relatively homogeneous data sets. Although the structure and content of the banks' records will differ to some degree, the distributed data sets all represent the same basic information: historical credit-card transactions. Privacy also is a concern for distributed, heterogeneous data. Different databases, containing different information, will have different levels of required security. Some may be too sensitive for any data mining, but others may allow limited querying for aggregate information. In some cases, privacy concerns may stem from a desire to profit from collected data. A company that has a lot of data may decide to issue subscriptions to a data-mining server, which provides aggregate information mined from their data. By allowing only limited querying, and by stipulating query restrictions in the subscription

contract, companies may be able to profit from the massive volume of data they collect routinely, while protecting individuals' privacy.

0.3.4 Is this view of DDM realistic or far-fetched?

These arguments notwithstanding, our current discussion would be incomplete if we did not address whether this view of DDM is realistic or far-fetched. With respect to mining relatively homogeneous databases for purposes of classification, the answer is easy. It is not far-fetched, and in fact Stolfo's group provides web-accessible software for distributed data mining (Stolfo 1998).

With respect to mining heterogeneous databases (containing completely different, but related, tables), there also is evidence that it is not far-fetched. At its most simple, heterogeneous DDM could involve SQL queries to auxiliary databases for what to this point I vaguely have called "aggregate statistics." Is this realistic? In other words, is it technically feasible for data mining to be performed through SQL queries for statistics, without access to the underlying data? The answer, of course, is "yes." Several authors discuss how to do just this. The main insight is that matching hypotheses against the data is not necessary: for most of the processing, all that is needed is a set of *sufficient statistics* from which the results of matching can be computed (Fayyad 1997). Separating the generation of the sufficient statistics from their use in the evaluation of hypotheses allows each to be treated separately—first using the data to populate the statistics data structure and then operating only on the data structure—which affords both optimized use of memory and improved run-time complexity (Aronis and Provost 1997). More specifically, consider mining classification models. For most of the critical data mining operations, such as choosing nodes when constructing decision trees, one must tally for all the instances (at a particular point in the search) the class labels associated with the different values of each attribute. A straightforward data structure to store such statistics is a contingency table of instance counts for each attribute, indexed by attribute-value and class. This data structure can be populated by SQL requests for statistics (Agrawal and Shim 1995; Agrawal and Shim 1996; John and Lent 1997; Graefe, Fayyad, and Chaudhuri 1998).

Sarawagi et al. (1998) provide perhaps the most comprehensive discussion of integrating data mining with database management systems. Their focus is on mining association rules, but they illustrate principles that apply more generally. They point to several efforts to extend SQL to support mining operations, and discuss expressing mining algorithms in SQL. In particular, Sarawagi et al. discuss pushing into the database system parts of the application program that perform intensive computations on the individual records, instead of bringing the records of the database into the application program. One method is to encapsulate the statistics gathering as a stored procedure, which is executed on the database machine. A somewhat different approach is to represent the individual data mining operations as user-defined functions placed in SQL data scan queries (which also will run on the database machine) (Agrawal and Shim 1995; Agrawal and Shim 1996). Sarawagi et al. also consider the more general case where a preprocessor translates data mining operations into the appropriate form for a particular environment.

There also has been work demonstrating the feasibility of distributed mining of heterogeneous data. Aronis et al. (1997) describe the WoRLD (Worldwide Relational Learning Daemon) system, that mines multiple, multitable databases distributed across networks. The key to the WoRLD's ability to treat distributed databases transparently is its use of spreading activation (Quillian 1968), instead of item-by-item matching, as the basic operation of the inductive engine. Each instance is labelled

with a marker, and the WoRLD propagates these markers through databases looking for features where markers of one class accumulate. This process can span several databases, possibly on different machines, with markers transmitted across network links. As with the WWW, there is no need for a master map of the entire structure—each database can have its own links to other related databases, which the WoRLD can follow as it encounters them.

0.4 Discussion

Besides being tedious, being prohibited by privacy restrictions, and potentially losing information, the process of coalescing already-distributed databases introduces other problems. The content of many databases is dynamic: it changes in response to changes in the world. Once data miners create their own mining database that incorporates information from auxiliary databases, the mining database quickly can become obsolete. Data miners do not want to, and often are not equipped to, duplicate the management of the auxiliary data. Neither do they want to rebuild their mining database repeatedly. A DDM system that can query the existing auxiliary databases as needed, leaving their management to their managers, would obviate this problem—also created by the awkward, non-distributed view.[4]

At this point, you may feel that I have glossed over a lot of very hard problems—and you would be right. As an example of a particularly telling problem, how is ontological mismatch to be resolved? Different databases may use the same term to mean different things, and may use different terms to refer to the same thing. A DDM system would have to either (partially) solve this problem, or would have to make very strong assumptions. I would counter this (correct) observation by pointing out that this is not a problem specific to distributed data mining; it just comes to the fore when it has to be automated. This is a hard, basic research problem for data mining, that *current* data mining research is glossing over.

0.4.1 The DDM view opens new avenues for research

Lots of other fundamental KDD research problems, that go beyond designing new or faster induction algorithms, also rear their attractive heads. Solutions to some technical DDM problems are beginning to be addressed (Grossman and Bailey 1998). For example, how can heterogeneous processors and network links best be used (Grossman, Bailey, Kasif, Mon, Ramu, and Malhi 1998)? However, consider a different type of problem. While building their monolithic database, data miners often notice that "database X is relevant in such-and-such a way," and work to incorporate X. Few authors address the fundamental issue: How and from where do such insights come? Taking a DDM view (and assuming the existence of database maps or metadata, other relevant research areas), systems themselves should be able to notice that "database X is relevant in such-and-such a way," and query it. Once again, the attempt to automate this part of the process brings it into the research spotlight.

DDM as simply issuing SQL queries, while currently the most feasible approach, certainly is not the most ambitious vision.

[4] At first it may seem that in such cases, managing the data in a distributed database management system (DDBMS), with incorporated mining routines, would be sufficient. However, although mining DDBMSs is one very interesting method for DDM, this view is incomplete. For organizational reasons, many auxiliary databases are not, and will not be, managed in a DDBMS; yet it still may be useful to use them when mining. Viewing DDM only as mining a DDBMS either (1) demands that a particular DDBMS incorporate all auxiliary data whenever they are deemed potentially relevant, or (2) restricts the use of some such data.

A DDM visionary might see distributed mining agents cooperating in a knowledge economy. Closer to feasibility are data mining servers, available on the network, that publish their capabilities for data mining clients to consider. These servers could sit atop data with varying degrees of privacy, and if necessary could serve only trusted clients or clients under contract. The emergence of a knowledge economy for data mining may not be all that far away. Once distributed data mining becomes a reality, it is not a stretch to foresee organizations developing that profit by providing knowledge services. They will store and manage large and potentially changing databases, and data miners will pay for access. For instance, fraud detectors and target marketers would be delighted if a demographic data provider were established, which for a small fee would provide just-the-right auxiliary knowledge for a particular problem.

0.4.2 DDM as knowledge discovery

Section 0.2 presented arguments for distributed data mining based on computational efficiencies due to parallelization of the distributed mining processes. I have tried to argue that although the effect of parallelization is important, it is less important than the effect of taking a distributed view of the problem. The question of whether the distributed mining processes must reside on separate processors, or whether they profitably could be simulated on a single processor, does not seem particularly important when juxtaposed with the conceptual neatness of the DDM problem formulation. Of all the examples heretofore presented, the most convincing in this respect may be DDM's potential to eliminate much of the manual effort needed to coalesce heterogeneous, multitable databases. It seems that, in light of the current state of the art, a data miner would value a reduction in the manual effort (of coalescence) more highly than the corresponding reduction in computational effort (by parallization).

There is another way in which a distributed view of data mining may lead to remarkable computational efficiencies. Data mining is one element of a larger process of discovery. Therefore, the correct direction and manner to proceed are inherently ill defined. A single explorer must try one thing, then another, then another, and so on, until either he finds something interesting or he runs out of resources. This describes the essential nature of the process of exploration, including not only what has been called "the knowledge discovery process," but also geographic exploration (e.g., in the fifteenth and sixteenth centuries) and the typical process of science. The length of a discovery's delay is related directly to the explorer's ability to prioritize possible paths of exploration.

On the other hand, a (distributed) group of explorers follows many paths simultaneously. Indeed, different explorers follow different paths almost by necessity. Herein lie two key insights. First, the speed of the group progresses at the speed of the fastest member of the group. Second, the entire group capitalizes on the discovery (once it is made known), and ratchets up its goals. Examples abound. In geographic discovery, once a shorter route was found to a desired destination, explorers used it to set their sights even higher for future discoveries. In science, once a (sub-)problem has been solved by one research group, the results are published and all research groups can now "stand on their shoulders" (paraphrasing Newton). Why is this relevant to distributed data mining? As with other explorations, by its very nature knowledge discovery is an ill-defined process. Thus we must try and try again to formulate just-the-right direction and manner of search. We may have many possibilities at the outset, but little reason to prefer one over another. If each is time consuming, discovery may be inefficient. However, every

weather-worn data miner has experienced the phenomenon that once the right problem formulation is concocted, discoveries are made remarkably, sometimes embarassingly, quickly.

If many different starting points and directions are begun simultaneously, and the "right one" is among the group, then the discovery may be made very quickly in real time, and perhaps even with much less combined effort than in the sequential case. In a different context, this phenomenon has been called a "combinatorial implosion" (Kornfeld 1982), and has been studied in other areas of artificial intelligence (e.g., for constraint satisfaction (Clearwater, Huberman, and Hogg 1991; Clearwater, Huberman, and Hogg 1992)).

0.5 Summary

Distributed data mining lately has been receiving increasing attention. Most work uses distributed processing to scale up to large databases, and the rich decomposability of data mining problems has led to successful techniques all along the spectrum from fine- to coarse-grained. The search space can be partitioned and different processors can search different parts, or the data can be partitioned. Nevertheless, scaling up is but one motivation for distributed data mining. Distributed data mining also eliminates the need to coalesce already distributed data. Coalescence is severly problematic, both in principle and in practice. Often, certain data are left out of a problem formulation not because they are deemed irrelevant, but because including them is too awkward. Distributed data mining avoids these problems, and also eliminates storage, time, and data management inefficiencies associated with coalescence. Finally, privacy concerns may prohibit coalescence altogether. Distributed data mining can allow access to a wide variety of data, while protecting data privacy.

Potential consumers of auxiliary knowledge already exist—consumers who are not shy about spending money on data mining. Potential suppliers await the development of infrastructure. Once DDM becomes a practical reality, implying not only solutions to the computational and network-related issues of distributed mining, but also the existence of published data maps and meta-data, then data-mining-supported knowledge economies will develop.

Acknowledgements

I thank John Aronis, Lars Asker, Bruce Buchanan, Jason Catlett, Scott Clearwater, Pedro Domingos, Phil Chan, Doug Fisher, Dan Hennessy, David Jensen, Ronny Kohavi, Rich Segal, Sal Stolfo, and anonymous referees of previous papers, who have influenced my view of scaling up and of DDM. Thanks also to the authors of the work surveyed, to the many who have pointed to relevant work, and especially to Venkat Kolluri, who has contributed in many ways.

Bibliography

Agrawal, R. and K. Shim (1995). Developing tightly-coupled applications on IBM DB2/CS relational database system: Methodology and experience. Research Report RJ 10005(89094), IBM Corporation.

Agrawal, R. and K. Shim (1996). Developing tightly-coupled data mining applications on a relational database system. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, Menlo Park, CA, pp. 287–290. AAAI Press.

Ali, K. M. and M. J. Pazzani (1996). Error reduction through learning multiple descriptions. *Machine Learning 24*(3), 173–202.

Aronis, J., V. Kolluri, F. Provost, and B. Buchanan (1997). The WoRLD: Knowledge discovery from multiple distributed databases. In *Proceedings of Florida Artificial Intelligence Research Symposium (FLAIRS-97)*.

Aronis, J. and F. Provost (1997). Increasing the efficiency of data mining algorithms with breadth-first marker propagation. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, Newport Beach, CA.

Blake, C., E. Keogh, and C. Merz (1998). UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Buchanan, B. and T. Mitchell (1978). Model-directed learning of production rules. In D. Waterman and F. Hayes-Roth (Eds.), *Pattern Directed Inference Systems*. New York, NY: Academic Press.

Buntine, W. (1991). *A theory of learning classification rules*. Ph. D. thesis, School of Computer Science, University of Technology, Sydney, Australia.

Catlett, J. (1991). *Megainduction: Machine learning on very large databases*. Ph. D. thesis, School of Computer Science, University of Technology, Sydney, Australia.

Chan, P. and S. Stolfo (1993). Toward parallel and distributed learning by meta-learning. In *Working Notes AAAI Workshop Knowledge Discovery in Databases*, pp. 227–240.

Chan, P. and S. Stolfo (1997). On the accuracy of meta-learning for scalable data mining. In *Journal of Intelligent Information Systems*, Volume 8, pp. 5–28.

Cheung, D., J. Han, V. Ng, A. W. Fu, and Y. Fu (1996). A fast distributed algorithm for mining association rules. In *Proc. 1996 Int'l Conf. on Parallel and Distributed Information Systems (PDIS'96)*, Miami Beach, Florida, USA., pp. 31–44.

Cheung, D., V. Ng, A. Fu, and Y. Fu (1996). Efficient mining of association rules in distributed databases. *IEEE Transaction on Knowledge and Data Engineering 8*(6), 911–922.

Clearwater, S., T. Cheng, H. Hirsh, and B. Buchanan (1989). Incremental batch learning. In *Proceedings of the Sixth International Workshop on Machine Learning*, San Mateo CA:, pp. 366–370. Morgan Kaufmann.

Clearwater, S. and F. Provost (1990). RL4: A tool for knowledge-based induction. In *Proceedings of the Second International IEEE Conference on Tools for Artificial Intelligence*, pp. 24–30. IEEE C.S.Press.

Clearwater, S. H., B. A. Huberman, and T. Hogg (1991, November 22). Cooperative solution of constraint satisfaction problems. *Science 254*, 1181–1183.

Clearwater, S. H., B. A. Huberman, and T. Hogg (1992). Cooperative problem solving. In B. A. Huberman (Ed.), *Computation: The Micro and the Macro View*, pp. 33–70. New Jersey: World Scientific.

Cook, D. and L. Holder (1990). Accelerated learning on the connection machine. In *Proceedings of the Second International IEEE Conference on Tools for Artificial Intelligence*, San Mateo CA, pp. 366–370. Morgan Kaufmann.

Cooper, G. F. and E. Herskovits (1992). A bayesian method for the induction of probabilistic networks from data. *Machine Learning 9*, 309–347.

Craven, M. W. (1996). *Extracting Comprehensible Models from Trained Neural Networks*. Ph. D. thesis, University of Wisconson – Madison. Technical Report No. 1326.

Danyluk, A. and F. Provost (1993). Small disjuncts in action: Learning to diagnose errors in the telephone network local loop. In P. Utgoff (Ed.), *Machine Learning: Proceedings of the Tenth International Conference*, pp. 81–88. Morgan Kaufmann Publishers, Inc.

DesJardins, M. and D. F. Gordon (1995). Special issue on bias evaluation and selection. *Machine Learning 20*(1/2).

Dietterich, T. G. (1997). Machine learning research: Four current directions. *AI Magazine 18(4)*, 97–136.

Domingos, P. (1996). Efficient specific-to-general rule induction. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, Menlo Park, CA, pp. 319–322. AAAI Press.

Domingos, P. (1997). Knowledge acquisition from examples via multiple models. In D. H. Fisher (Ed.), *Proceedings of the Fourteenth International Conference on Machine Learning (ICML-97)*, pp. 98–106. San Francisco, CA:Morgan Kaufmann.

Fayyad, U. (1997). Editorial. *Data Mining and Knowledge Discovery 1(1)*, 5–10.

Fayyad, U., N. Weir, and S. Djorgovski (1993). SKICAT: A machine learning system for automated cataloging of large scale sky surveys. In *Proceedings of the Tenth International Conference on Machine Learning*, pp. 112–119. Morgan Kaufmann.

Fayyad, U. M., G. Piatetsky-Shapiro, and P. Smyth (1996). From data mining to knowledge discovery: An overview. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthursamy (Eds.), *Advances in Knowledge Discovery and Data Mining*, pp. 1–34. Menlo Park, CA: AAAI Press.

Fox, E. A., R. M. Akscyn, R. Furuta, and J. Legsett (1995). *Communications of the ACM*, Volume 38(4). Morgan Kaufmann.

Freitas, A. and S. Lavington (1996). Using SQL primitives and parallel DB servers to speed up knowledge discovery in large relational databases. In *Cybernetics and Systems'96: Proceedings of the Thirteenth European Meeting on Cybernetics and Systems Research*, pp. 955–960.

Freitas, A. A. and S. H. Lavington (1997). *Mining Very Large Databases with Parallel Processing*. Norwell, MA: Kluwer Academic Publishers.

Frey, L. J. and D. H. Fisher (1999). Modeling decision tree performance with the power law. In D. Heckerman and J. Whittaker (Eds.), *Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics*. San Francisco, CA: Morgan Kaufmann.

Friedman, J. H. (1997). Data mining and statistics: What's the connection? In *Proceedings of the 29th Symposium on the Interface Between Computer Science and Statistics*.

Fürnkranz, J. (1998). Integrative windowing. *Journal of Artificial Intelligence Research 8*, 129–164.

Galal, G., D. J. Cook, and L. Holder (1999). Exploiting parallelism in a scientific discovery system to improve scalability. *Journal of the American Society for Information Science*. To appear.

Graefe, G., U. Fayyad, and S. Chaudhuri (1998). On the efficient gathering of sufficient statistics for classification of large SQL databases. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data-Mining*, New York, N.Y., pp. 204–208. AAAI Press.

Grossman, R. and S. Bailey (1998). A tutorial introduction to high performance data mining. Tutorial given at the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98).

Grossman, R., S. Bailey, S. Kasif, D. Mon, A. Ramu, and B. Malhi (1998). The preliminary design of Papyrus: A system for high performance, distributed data mining over clusters, meta-clusters and super-clusters. In *Working Notes of the KDD-97 Workshop on Distributed Data Mining*, pp. 37–43.

Guo, Y. and J. Sutiwaraphun (1998). Knowledge probing in distributed data mining. In *Working Notes of the KDD-97 Workshop on Distributed Data Mining*, pp. 61–69.

Haines, T. L. (1998). Private communication.

Hall, L. O., N. Chawla, and K. W. Bowyer (1998). Combining decision trees learned in parallel. In *Working Notes of the KDD-97 Workshop on Distributed Data Mining*, pp. 10–15.

Harris-Jones, C. and T. L. Haines (1997). Sample size and misclassification: Is more always better? Working Paper AMSCAT-WP-97-118, AMS Center for Advanced Technologies.

Haussler, D. (1988). Quantifiying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence 36*, 177–221.

Hershberger, D. and H. Kargupta (1999). Distributed multivariate regression using wavelet-based collective data mining. Technical Report Technical Report EECS-99-02, School of EECS, Washington State University.

Holte, R., L. Acker, and B. Porter (1989). Concept learning and the problem of small disjuncts. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, San Mateo, CA, pp. 813–818. Morgan Kaufmann.

Huber, P. (1997). From large to huge: A statistician's reaction to KDD and DM. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, Menlo Park, CA, pp. 304–308. AAAI Press.

Jensen, D. and P. R. Cohen (1999). Multiple comparisons in induction algorithms. *Machine Learning*. To appear.

John, G. and P. Langley (1996). Static versus dynamic sampling for data mining. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pp. 367–370. AAAI Press.

John, G. and B. Lent (1997). SIPping from the data firehose. In *Proceedings of Third International Conference on Knowledge Discovery and Data Mining*, Menlo Park, CA, pp. 199–202. AAAI Press.

Kargupta, H., E. Johnson, E. R. Sanseverino, B.-H. Park, L. D. Silvestre, and D. Hershberger (1998). Scalable data mining from distributed, vertically partitioned feature space using collective mining and gene expression based genetic algorithms. In *Working Notes of the KDD-97 Workshop on Distributed Data Mining*, pp. 70–91. http://www.eecs.wsu.edu/˜hillol/pubs/bodhi.ps.Z.

Kargupta, H., B. Park, D. Hershberger, and E. Johnson (2000). Collective data mining: A new perspective toward distributed data mining. In H. Kargupta and P. Chan (Eds.), *Advances in Distributed Data Mining*. AAAI Press. This volume. (Also available as School of EECS, Washington State University Technical Report EECS-99-01.).

Kearns, M. and S. Seung (1995). Learning from a population of hypotheses. *Machine Learning 18*, 255–276.

Kohavi, R. (1996). *Wrappers for Performance Enhancement and Oblivious Decision Graphs*. Ph. D. thesis, Dept. of Computer Science, Stanford University, Palo Alto, CA.

Kohavi, R. and G. John (1997). Wrappers for feature subset selection. *Artificial Intelligence 97(1-2)*, 273–324.

Kohavi, R. and D. Sommerfield (1995). Feature subset selection using the wrapper model: Overfitting and dynamic search space topology. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, Menlo Park, CA. AAAI Press.

Kornfeld, W. A. (1982). Combinatorially implosive algorithms. *Communications of the ACM 25*(10), 155–171.

Kufrin, R. (1997). Generating C4.5 production rules in parallel. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pp. 565–670. Menlo Park, CA:AAAI Press.

Kumar, V. and V. Rao (1987). Parallel depth-first search, part 2: Analysis. *International Journal of Parallel Programming 16*, 501–519.

Lam, W. and A. Segre (1997). Distributed data mining of probabilistic knowledge. In *Proceedings of the 17th International Conference on Distributed Computer Systems (ICDCS)*, pp. 178–185.

Lathrop, R., T. Webster, T. Smith, and P. Winston (1990). ARIEL: A massively parallel symbolic learning assistant for protein structure/function. In P. Winston and S. Shellard (Eds.), *AI at MIT: Expanding Frontiers*, Cambridge, MA. MIT Press.

Musick, R. (1998). Supporting large-scale computational science. Technical Report UCRL-ID-129903, Center for Applied Scientific Computing, Lawrence Livermore National Lab.

Oates, T. and D. Jensen (1997). The effects of training set size on decision tree complexity. In D. Fisher (Ed.), *Machine Learning: Proceedings of the Fourteenth International Conference*, pp. 254–262. Morgan Kaufmann.

Oates, T., M. Schmill, and P. Cohen (1997). Parallel and distributed search for structure in multivariate time series. In *Proceedings of the Ninth European Conference on Machine Learning*, pp. 191–198. Berlin:Springer-Verlag.

Prodromidis, A. L. and S. J. Stolfo (1998). Pruning meta-classifiers in a distributed data mining system. In *Working Notes of the KDD-97 Workshop on Distributed Data Mining*, pp. 22–30.

Provost, F. (1992). *Policies for the Selection of Bias in Inductive Machine Learning*. Ph. D. thesis, Department of Computer Science, University of Pittsburgh, Pittsburgh, PA.

Provost, F. and J. Aronis (1996). Scaling up inductive learning with massive parallelism. *Machine Learning 23*, 33–46.

Provost, F. and B. Buchanan (1995). Inductive policy: The pragmatics of bias selection. *Machine Learning 20*, 35–61.

Provost, F. and D. Hennessy (1994). Distributed machine learning: scaling up with coarse-grained parallelism. In *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*.

Provost, F. and D. Hennessy (1996). Scaling up: Distributed machine learning with cooperation. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Menlo Park, CA, pp. 74–79. AAAI Press.

Provost, F., D. Jensen, and T. Oates (1999). Efficient progressive sampling. In *Proceedings of the SIGKDD Fifth International Conference on Knowledge Discovery and Data Mining*.

Provost, F. and V. Kolluri (1997). Scaling Up inductive algorithms: An overview. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining.*, Menlo Park, CA, pp. 239–242. AAAI Press.

Provost, F. and V. Kolluri (1999). A survey of methods for scaling up inductive algorithms. *Data Mining and Knowledge Discovery 3*(2), 131–169.

Quillian, R. (1968). Semantic memory. In M. Minsky (Ed.), *Semantic Information Processing*. Cambridge, MA: MIT Press.

Quinlan, J. (1983). Learning efficient classification procedures and their application to chess endgames. In R. Michalski, C. J., and T. Mitchell (Eds.), *Machine Learning: An AI approach*, pp. 463–482. Los Altos, CA: Morgan Kaufmann.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.

Rao, V. and V. Kumar (1987). Parallel depth-first search, part 1: Implementation. *International Journal of Parallel Programming 16*, 479–499.

Sarawagi, S., S. Thomas, and R. Agrawal (1998). Integrating association rule mining with relational database systems: Alternatives and implications. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*.

Savasere, A., E. Omiecinski, and S. Navathe (1995). An efficient algorithm for mining association rules in large databases. In *Proceedings of Twenty-First International Conference on Very Large Data Bases*, pp. 432–444. Morgan Kaufmann.

Segal, R. and O. Etzioni (1994). Learning decision lists using homogeneous rules. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Menlo Park, CA, pp. 619–625. AAAI Press.

Shafer, J., R. Agrawal, and M. Mehta (1996). SPRINT: A scalable parallel classifier for data mining. In *Proceedings of the Twenty-Second International Conference on Very Large Data Bases*, Mumbai, India.

Shasha, D. (1997). PC4.5. http://merv.cs.nyu.edu:8001/~binli/plinda.

Sikora, R. and M. J. Shaw (1996). A computational study of distributed rule learning. *Information Systems Research 7*(2), 189–197.

Smyth, P. and R. Goodman (1992). An information theoretic approach to rule induction from databases. *IEEE Transactions on Knowledge and Data Engineering 4*(4), 301–316.

Stolfo, S. (1998). http://www.cs.columbia.edu/˜sal/JAM/PROJECT.

Stolfo, S., D. Fan, W. Lee, A. Prodromidis, and P. Chan (1997). Credit card fraud detection using meta-learning: Issues and initial results. In *Proceedings of the AAAI-97 Workshop on AI Approaches to Fraud Detection and Risk Management (AAAI Technical Report WS-97-07)*, Menlo Park: CA, pp. 83–90. AAAI Press.

Stolfo, S., A. Prodromidis, S. Tselepis, D. Fan, W. Lee, and P. Chan (1997). JAM: Java agents for meta-learning over distributed databases. In *Proceedings of the AAAI-97 Workshop on AI Approaches to Fraud Detection and Risk Management (AAAI Technical Report WS-97-07)*, Menlo Park: CA, pp. 91–98. AAAI Press.

Toivonen, H. (1996). Sampling large databases for association rules. In *Proceedings of the Twenty-fourth International Conference on Very Large Data Bases*.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM 27*(11), 1134–1142.

Webb, G. (1995). OPUS: An efficient admissible algorithm for unordered search. *Journal of Artificial Intelligence Research 3*, 383–417.

Williams, G. (1990). *Inducing and Combining Multiple Decision Trees*. Ph. D. thesis, Australian National University, Canberra, Australia.

Wu, X. and W. H. Lo (1998). Multi-layer incremental induction. In *Proceedings of the Fifth Pacific Rim International Conference on Artificial Intelligence*, pp. 24–32. Springer-Verlag.

Yamanishi, K. (1997). Distributed cooperative bayesian learning strategies. In *Proceedings of the Tenth Annual Conference on Computational Learning Theory*, pp. 250–262. ACM Press.

Zaki, M. (1998). *Scalable Data Mining for Rules*. Ph. D. thesis, Department of Computer Science, University of Rochester, Rochester, NY.

Zaki, M. J., C. Ho, and R. Agrawal (1999). Scalable parallel classification for data mining on shared memory multiprocessors. In *Proceedings of IEEE International Conference on Data Engineering*.

Zaki, M. J., S. Parthasarathy, W. Li, and M. Ogihara (1997). Evaluation of sampling for data mining of association rules. In *Proceedings of the Seventh International Workshop on Research Issues in Data Engineering.*