Scaling Up Inductive Algorithms: An Overview

Foster Provost
NYNEX Science and Technology
White Plains, NY 10604
foster@nynexst.com

Venkateswarlu Kolluri
University of Pittsburgh
Pittsburgh, PA, 15260
venkat@lis.pitt.edu

Abstract

This paper establishes common ground for researchers addressing the challenge of scaling up inductive data mining algorithms to very large databases, and for practitioners who want to understand the state of the art. We begin with a discussion of important, but often tacit, issues related to scaling up. We then overview existing methods, categorizing them into three main approaches. Finally, we use the overview to recommend how to proceed when dealing with a large problem and where future research efforts should be focused.

Introduction

Organizations are beginning to amass very large repositories of customer, operations, scientific, and other sorts of data. One of the primary, explicit challenges of the knowledge discovery and data mining community is the development of inductive learning algorithms that scale up to large data sets (Fayyad, Piatetsky-Shapiro, & Smyth 1996). This paper overviews the work done to date addressing this challenge.

We first address the meaning of "scaling up" and highlight important issues that are often tacit in published work. We then show similarities between existing methods by grouping them into three high-level approaches: build fast algorithms, partition the data and use a relational representation. For each of these approaches we briefly describe the constituent methods. Finally, we conclude with recommendations based on this high-level view.

We regret that because of space limitations, it is impossible in this paper to provide references to even a fraction of the relevant published work. Instead, we have made available a detailed survey and comprehensive bibliography as a technical report (Provost & Kolluri 1997).

Why scale up?

The most commonly cited reason for attempting to scale inductive methods up to massive data sets is based on the prevailing view of data mining as classifier learning. When learning classifiers, increasing the size of the training set typically increases the accuracy of the learned models (Catlett 1991). In many cases, the degradation in accuracy when learning from smaller samples stems from over-fitting due to the need to allow the program to learn *small disjuncts* (Holte 1989) or due to the existence of a large number of features describing the data. Large feature sets increase the size of the space of models; they increase the likelihood that, by chance, a learning program will find a model that fits the data well, and thereby increase the size of the example sets required (Haussler 1988).

Scaling up is also an issue in applications not concerned with predictive modeling, but with the discovery of interesting knowledge from large databases. For example, the ability to learn small disjuncts well is often of interest to scientists and business analysts, because small disjuncts often capture special cases that were unknown previously—the analysts often know the common cases (Provost & Aronis 1996). As with classifier learning, 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.

It should be clear that scaling up to very large data sets implies, in part, that fast learning algorithms must be developed. There are, of course, other motivations for fast learners. For example, interactive induction (Buntine 1991), in which an inductive learner and a human analyst interact in real time, requires very fast learning algorithms in order to be practical. Wrapper approaches, which for a particular problem and algorithm iteratively search for feature subsets or good parameter settings (Kohavi & Sommerfield 1995) (Provost & Buchanan 1995), also require very fast learners because such systems run the learning algorithms multiple times, evaluating them under different conditions.

SCALING METHODS		
Main Approach	**General Methods**	**Example Techniques**
Fast algorithm	Restricted model space	linear discriminant, perceptron, decision stump
	Powerful search heuristics	greedy, divide & conquer
		incremental reduced error pruning
		avoid grow-and-prune
		constraint-based search reduction
		dynamic search-space restructuring
	Optimized representation	efficient data structures
		bookkeeping strategies
	Parallelization	search-space parallelization
		parallel matching
Data partitioning	Instance sampling	random sampling
		duplicate compaction
		stratified sampling
		peepholing
		information-theoretic peepholing
	Feature sampling	use relevance knowledge
		use statistical indications
		use subset studies
	Process samples incrementally	independent multi-sample learning
		sequential multi-sample learning
		sequential feature selection
	Process samples concurrently	multiple models, pick best
		combining model descriptions
		cooperative learning
Relational representations	Use relational database	access via SQL queries
		utilize parallel database engine
		push computation into DBMS
		utilize distributed databases
	Use relational background knowledge	inductive logic programming
		efficient subset of ILP

Figure 1: Methods for Scaling Up Inductive Algorithms

What is "scaling up"?

For most scaling up scenarios the limiting factor of the dataset has been the number of examples. The related algorithmic question is: what is the growth rate of the algorithm's run time as the number of examples increases? Also important, but less visible in published work, is the number of attributes describing each example. For most published work on inductive algorithms, one million training examples with a couple dozen attributes is considered to be a very large data set (100Mbyte-1Gbyte range). Most algorithms work has been done by researchers from the machine learning community, who are accustomed to dealing with flat files and algorithms that run in minutes or seconds on a desktop platform. Practitioners from the database community are used to dealing with multi-gigabyte databases. Typically, data preprocessing techniques are used to reduce the size of the data set presented to algorithms by orders of magnitude.

As may be expected, time complexity analyses do not tell the whole story. Although seldom discussed, space considerations are crucial to scaling up—most importantly, the absolute size of the main memory with which the data mining computing platform is equipped. Almost all existing implementations of learning 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 continual virtual memory thrashing, the algorithm will not scale well.

Finally, the goal of the learning must be considered when one considers scaling up. Evaluating the effectiveness becomes complicated if a degradation in the quality of the learning is permitted. The vast majority of work on learning algorithms uses classification accuracy as the metric by which different algorithms are compared. Thus, we are most interested in methods that scale up without a substantial decrease in accuracy. For algorithms that mine regularities from the data for purposes other than classification, metrics should be devised by which the effectiveness can be measured (and compared) as the system scales up. Some researchers have proposed "interestingness" criteria (Srikant & Agrawal 1996).

Three approaches

Many diverse techniques have been proposed and implemented for scaling up inductive algorithms. The similarities among the techniques become apparent when they are categorized into three main approaches. In most cases, techniques from separate categories are independent and can be applied simultaneously. Figure 1 summarizes the general methods that make up each of the three broad approaches to scaling up inductive algorithms, and lists some specific techniques.

Build fast algorithms

The most straightforward approach to scaling up inductive learning is to produce more efficient algorithms. Of course, for very large problems, even a fast linear-time algorithm may not be *sufficient* for practical data mining. However, it is usually the case that for very large problems, even with sampling and feature selection, a fast algorithm is still *necessary.*

Inductive algorithms have been framed as searching through a space of models for a model that performs well with respect to some criteria (Simon & Lea 1973) (Mitchell 1982). This view partitions fast-algorithm design into two categories of methods. First, one can restrict the space of models to be searched, based on the straightforward principle that a small model space will be much faster to search than a large one. Often simple models perform well (Holte 1993). Second, for a large model space, one can develop powerful search heuristics, where "powerful" means that the heuristics are efficient, yet often find competitive models. It is also important to consider algorithm engineering optimizations such as efficient data structures and bookkeeping schemes. Finally, most data mining tasks are decomposable, so fast algorithms can be built by taking advantage of parallel processing.

Partition the data

The **data partitioning** approach involves breaking the data set up into subsets, learning from one or more of the subsets, and possibly combining the results, as illustrated in Figure 2. Subsets can be subsets of examples or subsets of features, corresponding to selecting rows or columns of a data table.

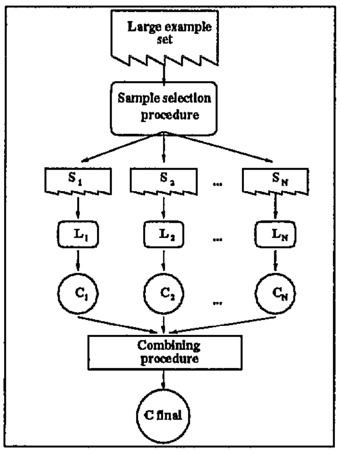

Figure 2: Learning using data partitioning.

Systems using a data partitioning approach select one or more subsets $S_1, \dots, S_n$ of the data based on a *selection procedure.* Learning algorithms $L_1, \dots, L_n$ are run on the corresponding subsets, producing concept descriptions $C_1, \dots, C_n$. Then the concept descriptions are processed by a *combining procedure,* which either selects from among $C_1, \dots, C_n$ or combines them, to produce a final concept description. The systems differ in the particular procedures used for selection and combining. They also differ in the amount and style of interaction among the learning algorithms and learned concept descriptions.

The most common approach for coping with the infeasibility of learning from very large data sets is a degenerate form of this model: select a single, small sample from the large data set. The differences between sampling techniques involve the selection procedure used, e.g., random or stratified sampling.

For taking advantage of more data, processing multiple subsets can be used to avoid thrashing by memory management systems. Also, if a learning algorithm's time complexity is more than linear in the number of examples, processing small, fixed-size samples sequentially can reduce the complexity to linear. Sequentially processing subsets allows the results of one data mining phase to feed into the next. Alternatively, the runs can be independent, and the class descriptions can be combined. In the case of independent runs, it is possible to use a system of distributed processors to mine the subsets concurrently.

Use a relational representation

The **relational representation** approach deals with cases when the data set cannot feasibly be represented as a flat file, because of its size. This is the case with any large relational database, as well as with other large relational structures such as those used for knowledge representation in artificial intelligence. In addition, for large databases the flattening-out process itself becomes quite time consuming, and keeping flat files around leads to the problems that relational databases are designed to avoid (e.g., update and delete anomalies).

One way to use relational data directly is to integrate data mining algorithms with database management systems (DBMSs). Integrated KDD/DBMS systems scale up by taking advantage of the storage efficiencies of relational representations, the use of indices, and the fact that DBMSs typically reside on powerful platforms that are optimized for database operations. Scaling can be extended further by making use of parallel database server technology.

Additional context can be added to any data mining problem by exploiting information contained in other databases. We must then ask whether all these databases must be resident on a single system, because concurrent analysis of different relations may give additional speedups. Moreover, a database of interest may be accessible over the network, but not practically transferrable. However, the ability to link together many databases further extends the scaling challenge.

Recommendations

When a data set fits in main memory, restricted model space learners should be tried first, because they are often effective at building competitive classifiers quickly. If the resulting simple classifiers are not satisfactory, there are several fast, effective algorithms for data sets that can fit into main memory.

When a data set does not fit in main memory, several techniques are clear choices when they apply. First, one should sample a subset of the data for training. A subset of the examples should be sampled, using stratified sampling when one class dominates strongly. A subset of the features should also be sampled, perhaps by doing empirical studies to determine relevance. Existing algorithms are fast enough that a good deal of experimentation (with data subsets) is possible to select the right problem parameters. Once a practitioner has chosen a good subset of examples, a good subset of features, and an algorithm with an efficient data representation, there may not be a significant increase in accuracy when learning with more data than will fit in main memory. Second, and perhaps too obvious to mention, if the computer's memory slots are not filled to capacity, a very cost-effective method for scaling up is to install more memory.

Once these straightforward methods have been exhausted, research results provide less guidance as to what approach to take. Parallel matching is very effective if one has access to a massively parallel machine. Taking advantage of powerful, well-tuned database systems via KDD/DBMS integration is a good idea if cycles on the database engine are readily available. Independent multi-sample learning shows promise for scaling up and retaining the flexibility of desktop data mining, and offers to take advantage of the large number of idle workstations that are already networked in most institutions. Unfortunately, there is currently a dearth of readily available technology.

Even when problem and environmental characteristics dictate a general approach, there is little guidance for choice among the various constituent techniques. For each approach, several techniques have been studied in isolation, but there exist few (if any) studies comparing their relative merits. For example, for partitioned data approaches, research has only just reached the border between the proof-of-concept stage and the comparative-evaluation stage. Both theoretical and empirical research is still needed before we can claim a thorough understanding.

We know even less about efficient methods for learning with massive amounts of relational knowledge. Given the storage economies possible with relational representations, their use promises that much larger data sets can be processed in main memory than with flat file representations. Thus, arguably the most promising direction for future scaling research is the development of fast, effective methods of mining relationally represented data. Such research would have broad implications, affecting the development of KDD/DBMS integrated systems, algorithms for learning in main memory, and partitioned data approaches.

Finally, we would like to stress again that many of these directions are still in the proof-of-concept stage. We should be careful not to assume any of these issues has been "solved" until we can point to studies that provide comprehensive comparisions of methods.

Acknowledgements

We are indebted to many, including John Aronis, Bruce Buchanan, Jason Catlett, Phil Chan, Pedro Domingos, Doug Fisher, Dan Hennessy, Ronny Kohavi, Rich Segal, Sal Stolfo, and many anonymous referees, who have influenced our views of scaling up. We also thank Tom Fawcett and Doug Metzler for comments on drafts, and Usama Fayyad for encouraging us to turn our informal summary into a formal survey. This work was partly supported by National Science Foundation grant IRI-9412549 and NCRR(NIH) grant RR06009.

References

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

Catlett, J. 1991. Megainduction: machine learning on very large databases. Ph.D. diss., Dept. of Computer Science, University of Sydney, Australia.

Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. 1996. Knowledge Discovery and Data Mining: Towards a Unifying Framework. In Proc. KDD-96, 82-88. AAAI Press.

Haussler, D. 1988. Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence*, 36, 177-221.

Holte, R.C. 1993. Very Simple Classification Rules Perform Well on Most Commonly Used Datasets. *Machine Learning*, 3, 63-91.

Kohavi, R. and Sommerfield, D. 1995. Feature subset selection using wrapper model: Overfitting and dynamic search space topology. In Proc. KDD-95.

Mitchell, T.M. 1982. Generalization as search. In *Artificial Intelligence*, 18(2), 203-226.

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

Provost, F.J. and Buchanan, B.G. 1995. Inductive Policy: The pragmatics of bias selection. *Machine Learning*, 20, 35-61.

Provost, F.J., and Kolluri, V. 1997. A Survey of Methods for Scaling Up Inductive Learning Algorithms, Technical Report ISL-97-3, Intelligent Systems Lab., University of Pittsburgh (http://www.pitt.edu/~uxkst/survey-paper.ps).

Srikant, R. and Agrawal, R. 1996. Mining Quantitative Association Rules in Large Relational Tables. In Proc. of the ACM SIGMOD Conf. on Mgmt. of Data, Montreal, Canada, June 1996.

Simon, H. A. and Lea, G. 1973. Problem solving and rule induction: A unified view. In Gregg (ed.), *Knowledge and Cognition*, 105-127. New Jersey: Lawrence Erlbaum Associates.