A Survey of Methods for Scaling Up
Inductive Algorithms

FOSTER PROVOST  provost@acm.org
Bell Atlantic Science and Technology, 500 Westchester Avenue, White Plains, New York 10604

VENKATESWARLU KOLLURI venkat@sis.pitt.edu
Department of Information Science, University of Pittsburgh, Pittsburgh, PA 15260, and Lycos, Inc.,
5001 Centre Avenue, Pittsburgh, PA 15213

Editor: Usama Fayyad

Abstract. One of the defining challenges for the KDD research community is to enable inductive learning
algorithms to mine very large databases. This paper summarizes, categorizes, and compares existing work on
scaling up inductive algorithms. We concentrate on algorithms that build decision trees and rule sets, in order to
provide focus and specific details; the issues and techniques generalize to other types of data mining. We begin
with a discussion of important issues related to scaling up. We highlight similarities among scaling techniques
by categorizing them into three main approaches. For each approach, we then describe, compare, and contrast
the different constituent techniques, drawing on specific examples from published papers. Finally, we use the
preceding analysis to suggest how to proceed when dealing with a large problem, and where to focus future
research.

Keywords: scaling up, inductive learning, decision trees, rule learning

1. Introduction

The knowledge discovery and data mining (KDD) community has challenged itself to
develop inductive learning algorithms that scale up to large data sets (Fayyad et al., 1996,
1996a; Piatetsky-Shapiro et al., 1996). This paper1 summarizes, categorizes, and compares
various existing methods. We restrict the survey’s scope to scalable algorithms, and do not
consider issues of efficient file system design, storage design, network interface design, or
problem formulation, except as they relate to the design of inductive algorithms. Although
we believe the categorization and lessons apply more generally, our analysis focuses prima-
riely on algorithms that build feature-vector-based classifiers (rather than those that include
structural or relational terms) in the form of decision trees or rule sets.

We first address the meaning of “scaling up” and highlight important issues. We then show
similarities between existing methods by grouping them into three high-level categories.
Within each category, we discuss the techniques themselves in some detail, showing the
similarities and differences between techniques of each type. Finally, we conclude with
suggestions for research and practice that emerge from the survey’s analysis.
2. Why scale up?

Organizations are amassing very large repositories of customer, operations, scientific, and other sorts of data. Fayyad et al. (1996b) cite several representative examples of databases containing many gigabytes (even terabytes) of data. KDD practitioners would like to be able to apply inductive learning algorithms to these large data sets in order to discover useful knowledge. The question of scalability asks whether the algorithm can process large data sets efficiently, while building from them the best possible models. However, the existence of very large data sets alone is not sufficient to motivate non-trivial scaling efforts. Why not just select a small subset of the data for data mining?

The most commonly cited reason for scaling up is that increasing the size of the training set often increases the accuracy of learned classification models (Catlett, 1991b). In many cases, the degradation in accuracy when learning from smaller samples stems from overfitting due to the need to allow the program to learn small disjuncts (Holte et al., 1989), elements of a class description that cover few data items. In some domains small disjuncts make up a large portion of the class description (Danyluk and Provost, 1993). In such domains, high accuracy depends on the ability to learn small disjuncts to account for these special cases. The existence of noise in the data further complicates the problem, because with a small sample it is impossible to tell the difference between a special case and a spurious data point.

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 example sets (Haussler, 1988). Things get particularly difficult when there are many features and there is the need to learn 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 may still be an issue. For example, the ability to learn small disjuncts well often is 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). 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 (Provost and Aronis, 1996).

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 practicable. Wrapper approaches, which for a particular problem and algorithm iteratively search for feature subsets or good parameter settings (Kohavi and Sommerfield, 1995;