Statistics Gathering for Learning from Distributed, Heterogeneous and Autonomous Data Sources

Doina Caragea, Jaime Reinoso, Adrian Silvescu, and **Vasant Honavar**
Artificial Intelligence Research Laboratory,
Computer Science Department, Iowa State University,
226 Atanasoff Hall, Ames, IA 50011-1040, USA,
{dcaragea, jreinoso, silvescu, and honavar }@cs.iastate.edu

Abstract

With the growing use of distributed information networks, there is an increasing need for algorithmic and system solutions for data-driven knowledge acquisition using distributed, heterogeneous and autonomous data repositories. In many applications, practical constraints require such systems to provide support for data analysis where the data and the computational resources are available. This presents us with *distributed learning* problems. We precisely formulate a class of distributed learning problems; present a general strategy for transforming traditional machine learning algorithms into distributed learning algorithms based on the decomposition of the learning task into *hypothesis generation* and *information extraction* components; formally defined the information required for generating the hypothesis (*sufficient statistics*); and show how to gather the sufficient statistics from distributed, heterogeneous, autonomous data sources, using a *query decomposition (planning)* approach. The resulting algorithms are *provably exact* in that the hypothesis constructed from distributed data is identical to that obtained by the corresponding algorithm when in the batch setting.

1 Introduction

Development of high throughput data acquisition technologies in a number of domains (e.g., biological sciences, environmental sciences, space sciences etc.) together with advances in digital storage, computing, and communications technologies have resulted in unprecedented opportunities for scientists, to utilize, at least in principle, the wealth of information available on the Internet in learning, scientific discovery, and decision making. In practice, effective use of the growing body of data, information, and knowledge to achieve fundamental advances in scientific understanding and decision making presents several challenges [Honavar *et al.*, 1998; 2001]:

- In such domains, data repositories are large in size, dynamic and physically distributed. Consequently, it is neither desirable nor feasible to gather all the data in a centralized location for analysis. Hence, efficient distributed learning algorithms that can operate across multiple autonomous data sources without the need to transmit large amounts of data are needed [Caragea *et al.*, 2001b; Davies and Edwards, 1999; Kargupta *et al.*, 1999; Prodromidis *et al.*, 2000; Provost and Kolluri, 1999].
- Data sources of interest are autonomously owned and operated. Consequently, the range of operations that can be performed on the data source (e.g., the types of queries allowed), and the precise mode of allowed interactions can be quite diverse (e.g., PROSITE repository of protein data limits queries to those that can be entered using the forms provided on the web). Hence, strategies for obtaining the required information within the operational constraints imposed by the data source are needed [Levy, 2000].
- Data sources are heterogeneous in structure (e.g., relational databases, flat files) and content (names and types of attributes and relations among attributes used to represent the data). For example, data about proteins include the amino acid sequences of proteins, multiple sources of 3-dimensional structures of proteins, multiple sources of structural features of proteins, multiple sources of protein-protein interaction data, multiple sources of functional annotations for proteins (according to different notions of protein function), among others.
- The ontologies implicit in the design of autonomous data sources (i.e., assumptions concerning *objects* that exist in the *world* , which determine the choice of *terms* and *relationships* among terms) often do not match the ontologies of the users of those data sources. In scientific discovery applications, because users often need to examine data in *different contexts* from *different perspectives*, methods for context-dependent dynamic information extraction from distributed data based on user-specified ontologies are needed to support information extraction and knowledge acquisition from heterogeneous distributed data ([Honavar *et al.*, 2001; Levy, 2000]).

Against this background, our main goal is to develop efficient strategies for extracting the information needed for learning (e.g., sufficient statistics) from heterogeneous, au-

tonomous, and distributed data sources, under a given set of ontological commitments in a given context.

The rest of the paper is organized as follows: In Section 2, we precisely formulate a class of distributed learning problems and present a general strategy for transforming traditional machine learning algorithms into distributed learning algorithms based on the decomposition of the learning task into *hypothesis generation* and *information extraction* components. The resulting algorithms are *provably exact* in that the hypothesis constructed from distributed data is identical to that obtained by the corresponding algorithm when in the batch setting. In Section 3, we formally define the *sufficient statistics* of a data set D with respect to a learning algorithm L and show how we can obtain these statistics from distributed data sets using a query decomposition (query planning) approach, assuming that data is presented to the algorithm as a table whose rows correspond to instances and whose columns correspond to attributes. Section 4 shows how heterogeneous data sources can be integrated and made to look as tables. Section 5 concludes with a summary and a brief outline of future research directions.

2 Distributed Learning

The problem of learning from distributed data sets can be summarized as follows: data is distributed across multiple sites and the learner's task is to discover useful knowledge from all the available data. For example, such knowledge might be expressed in the form of a decision tree or a set of rules for pattern classification. We assume that it is not feasible to transmit raw data between sites. Consequently, the learner has to rely on information (e.g., statistical summaries such as counts of data tuples that satisfy particular criteria) extracted from the sites.

Definition: A distributed learning algorithm L_D is said to be *exact* with respect to the hypothesis inferred by a learning algorithm L, if the hypothesis produced by L_D, using distributed data sets D_1 through D_n is the same (in terms of error) as that obtained by L when it is given access to the complete data set D, which can be constructed (in principle) by combining the individual data sets D_1 through D_n (i.e., $|error(L_D(D_1, \cdots, D_n)) - error(L(D_1 \cup \cdots \cup D_n))| = 0$)).

Definition: A distributed learning algorithm L_D is said to be *approximate* with respect to the hypothesis inferred by a learning algorithm L, if the hypothesis produced by L_D, using distributed data sets D_1 through D_n is a good approximation (in terms of error) of that obtained by L when it is given access to the complete data set D (i.e., $|error(L_D(D_1, \cdots, D_n)) - error(L(D_1 \cup \cdots \cup D_n))| < \epsilon, \forall \epsilon > 0)$).

Our approach to the exact/approximate distributed learning is based on a decomposition of the learning task into a *control part* which drives the execution of the algorithm toward the *generation of a hypothesis* and an *information extraction* part which is triggered by the control part whenever the algorithm requires *statistics* about the available data in order to generate the hypothesis (Figure 1).

In this approach to distributed learning, only the information extraction component has to effectively cope with the distributed and heterogeneous nature of the data in order to guarantee provably exact or approximate learning algorithms. The control component doesn't access data directly, but just the statistics extracted from data. These statistics are obtained from queries that the control part asks in the process of the hypothesis generation.

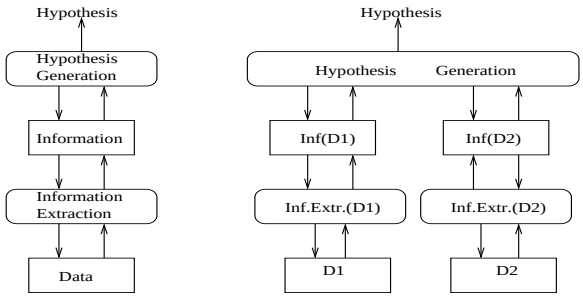

Figure 1: Task Decomposition into Hypothesis Generation and Information Extraction Components.

A query answering engine, which acts like a planner, decomposes the queries in terms of operators available to the data sources and returns the answers to the control part. The results of the queries can be seen as a statistics oracle which responds according to the needs of the control part (i.e., according to the requirements of a particular algorithm at each step). The classical example oracle, which here is distributed into a network, is used to provide sufficient statistics (e.g., counts for decision tree) to the centralized statistics oracle. The statistics oracle acts as a buffer between the data and the hypothesis corresponding to that data (Figure 2).

Our strategy for the distributed learning can be used to transform any batch learning algorithm into an efficient distributed learning algorithm, once the sufficient statistics with respect to that particular learning algorithm have been identified and a plan for gathering them has been found.

3 Sufficient Statistics

In order to be able to define sufficient statistics [Casella and Berger, 1990] in the context of learning, we look at a learning algorithm as search into a space of possible hypotheses. The hypotheses in the search space can be thought as defining a parametric function. A particular choice for the set of parameters will give us a particular hypothesis. In particular, those parameters can be estimated through learning based on the given data.

Definition: Let $\mathcal{F}$ be a class of functions that a learning algorithm is called upon to learn. A statistic $s(D)$ is called a *sufficient statistic* for learning a function $f \in \mathcal{F}$ given a data set $D = \{(x_1, y_1) \cdots, (x_m, y_m)\}$, if there exists a function g such that $g(s(D)) = f(D)$.

The particular learning algorithm considered determines a particular class of functions $\mathcal{F}$ (e.g., decision trees in the case of the decision tree algorithm, hyperplanes in the case of the SVM algorithm), which, in turn, determines a particular class of statistics (e.g., counts satisfying some criteria in the case of the decision trees). However, a sufficient statistic is defined with respect to a learning algorithm and a training data

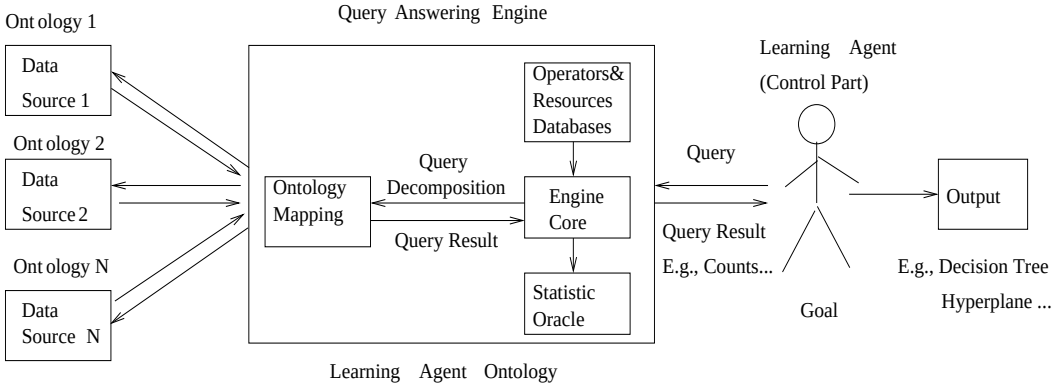

Figure 2: Distributed Learning System based on Task Decomposition into Hypothesis Generation and Information Extraction Components. The hypothesis generation component acts as a control part which triggers the execution of the information extraction component whenever the learning algorithm needs data. The information extraction component can be viewed as a query answering engine or planner.

set. Note that a data set D is a trivial sufficient statistic for D. However, whenever possible, we are interested in finding minimal sufficient statistics.

Definition: The sufficient statistic $s^*(D)$ is called *minimal sufficient statistic* if for any other sufficient statistic $s(D)$, $s^*(D)$ is a function of $s(D)$, i.e. there exists h such that $s^*(D) = h(s(D))$ for any data set D.

For some learning algorithms (e.g., Naive Bayes), the sufficient statistics have to be computed once in order for the learning algorithm to generate the hypothesis. However, for other algorithms (e.g., Decision Trees) the statistics gathering process and the hypothesis generation are interleaved. In this case, the target function cannot be computed in one step; the learner has to go back and forth between the data and the partially learned hypotheses several times. Instead of dealing with sufficient statistics, we use *partial sufficient statistics* here.

Examples of sufficient statistics for several classes of learning algorithms are shown below:

- Naive Bayes algorithm (NB) - counts of the instances that match certain criteria (e.g., attribute-value-class counts) represent minimal sufficient statistics.
- Decision Tree algorithm (DT) - counts of the instances that match certain criteria (e.g., attribute-value-class) are minimal partial sufficient statistics for computing one level of the tree. Subsequent counts that depend on the current built hypothesis (tree) are needed in order to find the final decision tree [Bhatnagar and Srinivasan, 1997].
- Support Vector Machines algorithm (SVM) - the weight vector which determines the separating hyperplane, can be considered sufficient statistics for the separating hyperplane.

3.1 Gathering Sufficient Statistics from Homogenous Data

To keep things simple, in what follows, we assume that regardless of the structure of the individual data repositories (relational databases, flat files, etc.) the effective data set for learning algorithm can be thought of as a table whose rows correspond to instances and whose columns correspond to attributes. We will show in Section 4 how heterogeneous data can be integrated and put into this form.

In designing distributed learning algoritms using our decomposition strategy, we assume that the identification of the sufficient statistics is done by a human expert, who also designs the control part of the algorithm (function g in the definition of the sufficient statistics). However, the gathering of the sufficient statistics necessary for learning (construction of the statistics oracle) is done automatically every time when the control part needs some statistics about data, by the query answering engine. The query answering engine receives as input a query, and builds a plan for this query according to the resources and the operators available to each data set (Figure 2).

The operators associated with a data source can be *primitive operators* (such as selection, projection, union, addition etc.), or they can be *aggregate operators* (e.g, counts or other database built-in functions). If the data source allows, the user can define *specific operators* (functions of the primitive operators). The set of primitive operators should be complete with respect to the set of learning tasks that needs to be executed (i.e., the application of these operators or functions of these operators is enough for gathering the information necessary for a particular learning task considered).

Definition: We call *learning plan* for computing the statistics s required by a learning algorithm L, from the distributed data sets $D_1, \cdots, D_n$ a procedure P, which transforms a given query into an execution plan. An *execution plan* can be seen as an expression tree, where each node corresponds to an operator and each leaf corresponds to basic statistics that can be extracted directly from the data sources. All the statistics that cannot be extracted directly from the data sources, should be replaced by their definitions recursively, until we obtain a plan in which only basic statistics appear as leaves.

Each of the operators available at the distributed data sources has a *cost* associated with it. Based on these operators and their costs, the query answering engine, which plays

the role of the planner, finds the best *execution plan* for the current query and sends it to the distributed data sources for execution. Each data source returns the statistics (answers to queries) extracted from its data to the query answering engine, which sends the final result to the learning agent. If the algorithm needs more information about data in order to finish its job, a new query is sent to the query answering engine, and the process repeats.

Definition: We say that two learning plans P_1 and P_2 are *equivalent* if they compute the same set of statistics. If we consider the costs associated with the operators included in a plan, we say that a learning plan P_1 is *more efficient* than an equivalent learning plan P_2 if the cost of the first plan is smaller than the cost of the second plan.

The job of the query answering engine is to find the best learning plan for a query, given a set of primitive operators, aggregate operators and user defined functions that can be applied to these operators, and their associated costs.

4 Gathering Sufficient Statistics from Heterogeneous Data Sources

In the previous section, we assumed that data is presented to the distributed algorithms as tables whose rows correspond to instances and whose columns correspond to attributes. However, in a heterogeneous environment, it is not trivial to get the data into this format. The differences in ontological commitments assumed by autonomous data sources present a significant hurdle in the flexible use of data from multiple sources and from different perspectives in scientific discovery.

To address this problem, we developed the INDUS software environment [Reinoso-Castillo, 2002] for rapid and flexible assembly of data sets derived from multiple data sources. INDUS is designed to provide a unified query interface over a set of distributed data sources which enables us to view each data source *as if* it were a table. Thus, a scientist can integrate data from different sources from his or her own perspective using INDUS. INDUS builds on a large body of work on information integration, including in particular, approaches to querying heterogeneous information sources using source descriptions [Levy *et al.*, 1996; Levy, 2000; Ullman, 1997], as well as distributed computing [Honavar *et al.*, 1998; Wong *et al.*,].

The input from a typical user (scientist) includes: an ontology that links the various data sources from the users point of view, executable code that performs specific computations, needed if they are not directly supported by the data sources, and a query expressed in terms of the user-specified ontology. In this case, the query answering engine, receives this query as input, finds the best execution plan for it, translates the plan according to the ontologies specific to the distributed data sources, and sends it to the distributed data sources. Each data source returns answers to the queries it receives and the planner translates them back to the user (learning agent) ontology. Thus, the user can extract and combine data from multiple data sources and store the results in a relational database which is structured according to his or her own ontology. The results of the queries thus executed are stored in a relational database and can be manipulated using application programs or relational database (SQL) operations and used to derive other data sets (as those necessary for learning algorithms).

More precisely, INDUS integration system is based on a federated query centric database approach [Mena *et al.*, 2000]. It consists of three principal layers which together provide a solution to the data integration problem in a scientific discovery environment (Figure 3):

- The *physical layer* allows the system to communicate with data sources. This layer is based on a federated database architecture (data is retrieved only in response to a query). It implements a set of *instantiators* which allow the interaction with each data source according to the constraints imposed by their autonomy and limited query capabilities. As a consequence, the central repository can view disparate data sources as if they were a set of tables (relations). New iterators may be added to the system when needed (e.g. a new kind of data source has become available, the functionality offered by a data source has changed).
- The *ontological layer* permits users to define one or more global (user/learning agent) ontologies and also to automatically transform queries into execution plans using a *query-centric* approach. More specifically, it consists of the following:
  - A meta-model, developed under a relational database, allowing users to define one or more global ontologies of any size. New statistics can be defined in terms of existing statistics using a set of compositional operators. The set of operators is extensible allowing users to add new operators as needed.
  - An interface for defining queries based on statistics in a global ontology.
  - An algorithm, based on a query-centric approach, for transforming those queries into an executable plan. The query-centric approach allows users to define each compound statistics in terms of basic statistics, (i.e. statistics whose instances are directly retrieved from data sources), using a predefined set of compositional operations. Therefore, the system has a description of how to obtain the set of instances of a global statistic based on extracting and processing instances from data sources. The plans describe what information to extract from each data source and how to combine the results.
  - An algorithm that executes a plan by invoking the appropriate set of instantiators and combining the results.
  - A repository for materialized (instantiated) plans which allow users to inspect them if necessary.
- Finally, the *user interface layer* enable users to interact with the system, define ontologies, post queries and receive answers. Also, the materialization of an executed plan can be inspected.

In conclusion, the most important features that INDUS offers in terms of data integration are summarized below:

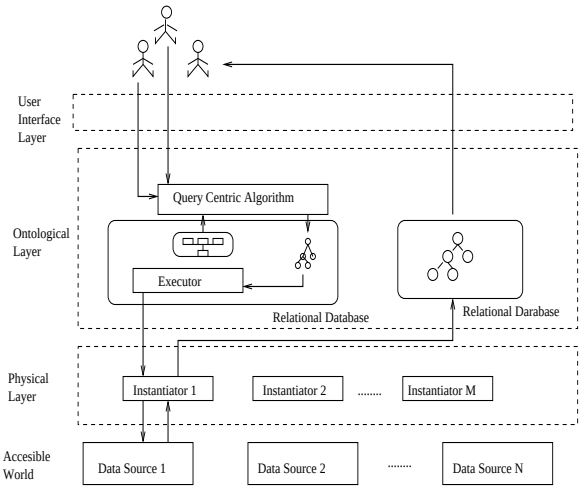

Figure 3: INDUS Information Integration Component.

- From a user perspective, data accessible through INDUS can be represented as tables in which the rows correspond to instances and columns correspond to attributes, regardless of the structure of the underlying data sources.
- The system can include several ontologies at any time. Individual users can introduce new ontologies as needed.
- New data sources can be incorporated into INDUS by specifying the data source descriptions including the corresponding data-source specific ontology and the set of instantiators.

The main difference between INDUS and other integration systems [Garcia-Molina *et al.*, 1997; Arens *et al.*, 1993; Knoblock *et al.*, 2001; Subrahmanian *et al.*, June 2000; Draper *et al.*, 2001; Paton *et al.*, 1999] is that it can include several ontologies at any time. The users can introduce new ontologies or add new data sources and their associated ontologies, as needed.

INDUS has been successfully used by computational biologists (including graduate and undergraduate students with varying degrees of expertise in computing) in our lab for quickly extracting and assembling the necessary data sets from multiple data repositories for exploring and visualizing protein sequence-structure-function relationships [Wang *et al.*, 2002; Andorf *et al.*, 2002].

For the purpose of distributed learning, INDUS is used to execute queries whose results are tables containing data of interest (e.g., counts). Depending on the operations that are allowed at a particular data source, these tables may contain raw but integrated data (according to the global ontology that different data sources share) or counts or other statistics extracted from the data sources. Thus, if a particular data source can answer specific queries, but it does not allow the execution of any program or the storage of any data at that site, then the answer to the query will produce a table that needs to be stored locally, but closely to the original data source, in order to avoid the transfer of large amount of data. This table can be further used to obtain the statistics needed for the generation of the hypothesis. On the other hand, if the data sources support aggregate operations (e.g., those that provide statistics needed by the learning algorithm) or allow user-supplied programs to be executed at the data source, we can avoid shipping large amounts of data from distributed repositories.

5 Summary and Future Work

The approach to the distributed learning taken in this paper is a generalization of a federated query centric database approach [Mena *et al.*, 2000]. In the case of distributed learning the set of operators is usually a superset of the operators used in classical databases. Besides, here the whole query answering process is just one step in the execution of the learning algorithm during which the statistics required by the algorithm are provided to the statistic oracle.

Assuming that the points of interaction between a learning algorithm and the available training data can be identified, the distributed learning strategy described here can be easily used to transform any batch learning algorithm into an exact (or at least approximate) distributed learning algorithm.

Future work is aimed at:

- Experiment with the decomposition strategy for various classes of learning algorithms and prove theoretical results with respect to the exact or the approximate quality of the distributed algorithms obtained.
- A big variety of data mining algorithms, such as decision trees, instance-based learners, Bayesian classifiers, Bayesian networks, multi-layer neural networks and support vector machines, among others, will be incorporated in our distributed learning system. Some of these algorithms can be easily decomposed into hypothesis generation and information extraction components according to our task decomposition strategy (e.g., Naive Bayes, decision trees), while others require substantial changes to the traditional learning algorithm (e.g., Support Vector Machines), resulting sometimes in new learning algorithms for distributed learning [Caragea *et al.*, 2001a; 2000].
- Formally define the set of operators for the algorithms that will be included in our distributed learning system and prove its completeness with respect to these algorithms.
- Extend the formal definitions for plans, formulate properties of these plans and prove these properties under various assumptions made in a distributed heterogeneous environment.

Acknowledgments

This work has been supported in part by grants from the National Science Foundation (#9982341, #9972653, #0219699), Pioneer Hi-Bred, Inc., Iowa State University Graduate College, and IBM.

References

[Andorf *et al.*, 2002] C. Andorf, D. Dobbs, and V. Honavar. Protein function classifiers based on reduced alphabet representations of protein sequences. 2002.

[Arens *et al.*, 1993] Y. Arens, C. Chin, C. Hsu, and C. Knoblock. Retrieving and integrating data from multiple information sources. *International Journal on Intelligent and Cooperative Information Systems*, 2(2):127–158, 1993.

[Bhatnagar and Srinivasan, 1997] R. Bhatnagar and S. Srinivasan. Pattern discovery in distributed databases. In *Proceedings of the AAAI-97 Conference*, 1997.

[Caragea *et al.*, 2000] D. Caragea, A. Silvescu, and V. Honavar. Agents that learn from distributed dynamic data sources. In *Proceedings of the Workshop on Learning Agents, Agents 2000/ECML 2000*, pages 53–61, Barcelona, Spain, 2000.

[Caragea *et al.*, 2001a] D. Caragea, A. Silvescu, and V. Honavar. Decision tree learning from distributed data. Technical Report TR, Iowa State University, Ames, IA, 2001.

[Caragea *et al.*, 2001b] D. Caragea, A. Silvescu, and V. Honavar. Invited chapter. toward a theoretical framework for analysis and synthesis of agents that learn from distributed dynamic data sources. In *Emerging Neural Architectures Based on Neuroscience*. Berlin: Springer-Verlag, 2001.

[Casella and Berger, 1990] G. Casella and R.L. Berger. *Statistical Inference*. Duxbury Press, Belmont, CA, 1990.

[Davies and Edwards, 1999] W. Davies and P. Edwards. Dagger: A new approach to combining multiple models learned from disjoint subsets. In *ML99*, 1999.

[Draper *et al.*, 2001] Denise Draper, Alon Y. Halevy, and Daniel S. Weld. The nimble XML data integration system. In *ICDE*, pages 155–160, 2001.

[Garcia-Molina *et al.*, 1997] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, V. Vassalos, and J. Widom. The tsimmis approach to mediation: Data models and languages. *Journal of Intelligent Information Systems, Special Issue on Next Generation Information Technologies and Systems*, 8(2), 1997.

[Honavar *et al.*, 1998] V. Honavar, L. Miller, and J.S. Wong. Distributed knowledge networks. In *Proceedings of the IEEE Conference on Information Technology*, Syracuse, NY, 1998.

[Honavar *et al.*, 2001] H. Honavar, A. Silvescu, D. Caragea, C. Andorf, J. Reinoso-Castillo, and D. Dobbs. Ontology-driven information extraction and knowledge acquisition from heterogeneous, distributed, autonomous biological data sources. In *Proceedings of the IJCAI-2001 Workshop on Knowledge Discovery from Heterogeneous, Distributed, Autonomous, Dynamic Data and Knowledge Sources*, Seattle, WA, 2001.

[Kargupta *et al.*, 1999] H. Kargupta, B.H. Park, D. Hershberger, and E. Johnson. Collective data mining: A new perspective toward distributed data mining. In H. Kargupta and P. Chan, editors, *Advances in Distributed and Parallel Knowledge Discovery*. MIT/AAAI Press, 1999.

[Knoblock *et al.*, 2001] Craig A. Knoblock, Steven Minton, Jose Luis Ambite, Naveen Ashish, Ion Muslea, Andrew Philpot, and Sheila Tejada. The ariadne approach to web-based information integration. *International Journal of Cooperative Information Systems*, 10(1-2):145–169, 2001.

[Levy *et al.*, 1996] A. Levy, A. Rajaraman, and J. Ordille. Querying heterogeneous information sources using source descriptions. 1996.

[Levy, 2000] A. Levy. *Logic-based techniques in data integration*. Kluwer Publishers, 2000.

[Mena *et al.*, 2000] E. Mena, A. Illarramendi, V. Kashyap, and A. Sheth. Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. *International journal on Distributed And Parallel Databases (DAPD)*, 8(2):223-272, 2000.

[Paton *et al.*, 1999] N.W. Paton, R. Stevens, P.G. Baker, C.A. Goble, and S. Bechhofer. Query processing in the tambis bioinformatics source integration system. In *Proc. 11th Int. Conf. on Scientific and Statistical Databases (SSDBM)*, pages 138–147. IEEE Press, 1999.

[Prodromidis *et al.*, 2000] A.L. Prodromidis, P. Chan, and S.J. Stolfo. Meta-learning in distributed data mining systems: Issues and approaches. In H. Kargupta and P. Chan, editors, *Advances of Distributed Data Mining*. AAAI Press, 2000.

[Provost and Kolluri, 1999] Foster J. Provost and Venkateswarlu Kolluri. A survey of methods for scaling up inductive algorithms. *Data Mining and Knowledge Discovery*, 3(2):131–169, 1999.

[Reinoso-Castillo, 2002] J. Reinoso-Castillo. Ontology-driven query-centric federated solution for information extraction and integration from autonomous, heterogeneous, distributed data sources, 2002.

[Subrahmanian *et al.*, June 2000] V.S. Subrahmanian, P. Bonatti, J. Dix, T. Eiter, S. Kraus, F. Ozcan, and R. Ross. *Heterogeneous Agent Systems: Theory and Implementation*. MIT Press, June 2000.

[Ullman, 1997] J. Ullman. Information integration using logical views. volume 1186, pages 19–40, Berlin: Springer-Verlag, 1997.

[Wang *et al.*, 2002] X. Wang, D. Schroeder, D. Dobbs, , and V. Honavar. Data-driven discovery of rules for protein function classification based on sequence motifs: Rules discovered for peptidase families based on meme motifs outperform those based on prosite patterns and profiles. 2002.

[Wong *et al.*,] J. Wong, G. Helmer, V. Honavar, V. Naganathan, S. Polavarapu, , and L. Miller. Smart mobile agent facility. *Journal of Systems and Software*, 56:9–22.