Decision Tree Induction from Distributed Heterogeneous Autonomous Data Sources

Doina Caragea, Adrian Silvescu, and Vasant Honavar

Artificial Intelligence Research Laboratory,
Computer Science Department, Iowa State University,
226 Atanasoff Hall, Ames, IA 50011-1040, USA
{dcaragea,silvescu,honavar}@cs.iastate.edu

Summary. 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; and demonstrate the application of this strategy to devise algorithms for decision tree induction (using a variety of splitting criteria) from distributed data. The resulting algorithms are *provably exact* in that the decision tree constructed from distributed data is identical to that obtained by the corresponding algorithm when in the batch setting. The distributed decision tree induction algorithms have been implemented as part of INDUS, an agent-based system for data-driven knowledge acquisition from heterogeneous, distributed, autonomous data sources.

1 Introduction

Recent advances in computing, communications, and digital storage technologies, together with development of high throughput data acquisition technologies have made it possible to gather and store large volumes of data in digital form. For example, advances in high throughput sequencing and other data acquisition technologies have resulted in gigabytes of DNA and protein sequence data, and gene expression data being gathered at steadily increasing rates in biological sciences. These developments have resulted in unprecedented opportunities for large-scale data-driven knowledge acquisition with the potential for fundamental gains in scientific understanding (e.g., characterization of macromolecular structure-function relationships in biology) in many data-rich domains. In such applications, the data sources of interest are typically physically distributed, heterogenuous, and are often autonomous.

Given the large size of these data sets, gathering all of the data in a centralized location is generally neither desirable nor feasible because of bandwidth and storage requirements. In such domains, there is a need for knowledge acquisition systems that can perform the necessary analysis of data at the locations where the data and the computational resources are available and transmit the results of analysis (knowledge acquired from the data) to the locations where they are needed. In other domains, the ability of autonomous organizations to share raw data may be limited due to a variety of reasons (e.g., privacy considerations). In such cases, there is a need for knowledge acquisition algorithms that can learn from statistical summaries of data (e.g., counts of instances that match certain criteria) that are made available as needed from the distributed data sources in the absence of access to raw data. Furthermore, data sources may be heterogeneous in structure (e.g., relational databases, flat files) and/or content (e.g., names and types of attributes and relations among attributes used to represent the data). Data-driven learning occurs within a context, or under certain ontological commitments on the part of the learner. In most applications of machine learning, the ontological commitments are implicit in the design of the data set. However, when several independently generated and managed data repositories are to be used as sources of data in a learning task, the ontological commitments that are implicit in the design of the data repositories may or may not correspond to those of the user in a given context. Hence, methods for context-dependent dynamic information extraction and integration from distributed data based on user-specified ontologies are needed to support knowledge aquisition from heterogeneous distributed data [1, 2].

Against this background, this paper presents an approach to the design of systems for learning from distributed, autonomous and heterogeneous data sources. We precisely formulate a class of distributed learning problems; present a general strategy for transforming a large class of traditional machine learning algorithms into distributed learning algorithms; and demonstrate the application of this strategy to devise algorithms for decision tree induction (using a variety of splitting criteria) from distributed data. The resulting algorithms are *provably exact* in that the decision tree constructed from distributed data is identical to that obtained by the corresponding algorithm in the batch setting (i.e., when all of the data is available in a central location).

Agent-oriented software engineering [3, 1] offers an attractive approach to implementing modular and extensible distributed computing systems. For the purpose of this discussion, an agent is an encapsulated information processing system that is situated in some environment and is capable of flexible, autonomous action within the constraints of the environment so as to achieve its design objectives. The distributed decision tree induction algorithms described in this paper have been implemented as part of INDUS (Intelligent Data Understanding System), an agent-based system for data-driven knowledge acquisition from heterogeneous, distributed, autonomous data sources.

2 Distributed Learning

To keep things simple, in what follows, we will 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 later discuss how heterogeneous data can be integrated and put in this form. In this setting, the problem of learning from distributed data sets can be summarized as follows: The data is distributed across multiple autonomous sites and the learner's task is to acquire useful knowledge from this data. For instance, such knowledge might take the form of a decision tree or a set of rules for pattern classification. In such a setting learning can be accomplished by an agent that visits the different sites to gather the information needed to generate a suitable model (e.g., a decision tree) from the data (*serial distributed learning*). Alternatively, the different sites can transmit the information necessary for constructing the decision tree to the learning agent situated at a central location (*parallel distributed learning*). 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. Our approach to learning from distributed data sets involves identifying the information requirements of existing learning algorithms, and designing efficient means of providing the necessary information to the learner while avoiding the neeed to transmit large quantities of data.

2.1 Exact Distributed Learning

We say that a *distributed learning* algorithm L_d (e.g., for decision tree induction from distributed data sets) is *exact* with respect to the hypothesis inferred by a batch learning algorithm L (e.g., for decision tree induction from a centralized data set) if the hypothesis produced by L_d using distributed data sets $D_1, \cdots, D_n$ stored at sites $1, \cdots, n$ (respectively), is the same as that obtained by L from the complete data set D obtained by appropriately combining the data sets $D_1, \cdots, D_n$. Similarly, we can define exact distributed learning with respect to other criteria of interest (e.g., expected accuracy of the learned hypothesis). More generally, it might be useful to consider *approximate* distributed learning in similar settings. However, the discussion that follows is focused on exact distributed learning.

2.2 Horizontal and Vertical Data Fragmentation

In many applications, the data set consists of a set of tuples where each tuple stores the values of relevant attributes. The distributed nature of such a data set can lead to at least two common types of data fragmentation: *horizontal fragmentation* wherein subsets of data tuples are stored at different

sites; and *vertical fragmentation* wherein subtuples of data tuples are stored at different sites. Assume that a data set D is distributed among the sites $1, \cdots, n$ containing data set fragments $D_1, \cdots, D_n$. We assume that the individual data sets $D_1, \cdots, D_n$ collectively contain enough information to generate the complete dataset D.

2.3 Transforming Batch Learning Algorithms into Exact Distributed Learning Algorithms

Our general strategy for transforming a batch learning algorithm (e.g., a traditional decision tree induction algorithm) into an exact distributed learning algorithm involves identifying the information requirements of the algorithm and designing efficient means for providing the needed information to the learning agent while avoiding the need to transmit large amounts of data. Thus, we decompose the distributed learning task into *distributed information extraction* and *hypothesis generation* components.

The feasibility of this approach depends on the information requirements of the batch algorithm L under consideration and the (time, memory, and communication) complexity of the corresponding distributed information extraction operations.

In this approach to distributed learning, only the information extraction component has to effectively cope with the distributed nature of data in order to guarantee provably exact learning in the distributed setting in the sense discussed above. Suppose we decompose a batch learning algorithm L in terms of an information extraction operator I that extracts the necessary information from data set and a hypothesis generation operator H that uses the extracted information to produce the output of the learning algorithm L. That is, $L(D) = H(I(D))$. Suppose we define a distributed information extraction operator I_d that generates from each data set D_i, the corresponding information $I_d(D_i)$, and an operator C that combines this information to produce $I(D)$. That is, the information extracted from the distributed data sets is the same as that used by L to infer a hypothesis from the complete dataset D. That is, $C[I_d(D_1),\ I_d(D_2), \cdots, I_d(D_n)] = I(D)$. Thus, we can guarantee that $L_d(D_1, \cdots, D_n) = H(C[I_d(D_1), \cdots, I_d(D_n)])$ will be exact with respect to $L(D) = H(I(D))$.

3 Decision Tree Induction from Distributed Data

Decision tree algorithms [4, 5, 6] represent a widely used family of machine learning algorithms for building pattern classifiers from labeled training data. They can also be used to learn associations among different attributes of the data. Some of their advantages over other machine learning techniques include their ability to: select from all attributes used to describe the data, a subset of attributes that are relevant for classification; identify complex predictive

relations among attributes; and produce classifiers that can be translated in a straightforward manner, into rules that are easily understood by humans. A variety of decision tree algorithms have been proposed in the literature. However, most of them select recursively, in a greedy fashion, the attribute that is used to partition the data set under consideration into subsets until each leaf node in the tree has uniform class membership.

The ID3 (Iterative Dichotomizer 3) algorithm proposed by Quinlan (Quinlan, 1986) and its more recent variants represent a widely used family of decision tree learning algorithms. The ID3 algorithm searches in a greedy fashion, for attributes that yield the maximum amount of information for determining the class membership of instances in a training set S of labeled instances. The result is a decision tree that correctly assigns each instance in S to its respective class. The construction of the decision tree is accomplished by recursively partitioning S into subsets based on values of the chosen attribute until each resulting subset has instances that belong to exactly one of the M classes. The selection of attribute at each stage of construction of the decision tree maximizes the estimated expected information gained from knowing the value of the attribute in question.

Different algorithms for decision tree induction differ from each other in terms of the criterion that is used to evaluate the splits that correspond to tests on different candidate attributes. The choice of the attribute at each node of the decision tree greedily maximizes (or minimizes) the chosen splitting criterion. To keep things simple, we assume that all the attributes are discrete or categorical. However, all the discussion below can be easily generalized to continuous attributes.

Often, decision tree algorithms also include a pruning phase to alleviate the problem of overfitting the training data. For the sake of simplicity of exposition, we limit our discussion to decision tree construction without pruning. However, it is relatively straightforward to modify the proposed algorithms to incorporate a variety of pruning methods.

3.1 Splitting Criteria

Some of the popular splitting criteria are based on entropy [4] which is used by Quinlan's ID3 algorithm and its variants, and the Gini Index [5] which is used by Breiman's CART algorithm, among others. More recently, additional splitting criteria that are useful for exploratory data analysis have been proposed [6].

Consider a set of instances S which is partitioned into M disjoint subsets (classes) $C_1, C_2, ..., C_M$ such that $S = \bigcup_{i=1}^{M} C_i$ and $C_i \bigcap C_j = \emptyset \; \forall i \neq j$. The estimated probability that a randomly chosen instance $s \in S$ belongs to the class C_j is $p_j = \frac{|C_j|}{|S|}$, where $|X|$ denotes the cardinality of the set X. The estimated *entropy* of a set S measures the expected information needed to identify the class membership of instances in S, and is defined as follows:

$entropy(S) = -\sum_j \frac{|C_j|}{|S|} \cdot \log_2 \left(\frac{|C_j|}{|S|} \right)$. The estimated *Gini index* for the set S containing examples from M classes is defined as follows: $gini(S) = 1 - \sum_j (\frac{|C_j|}{|S|})^2$. Given some impurity measure (either the entropy or Gini index, or any other measure that can be defined based on the probabilities p_j) we can define the estimated *information gain* for an attribute a, relative to a collection of instances S as follows: $IGain(S, a) = I(S) - \sum_{v \in Values(a)} \frac{|S_v|}{|S|} I(S_v)$ where $Values(A)$ is the set of all possible values for attribute a, S_v is the subset of S for which attribute a has value v, and $I(S)$ can be $entropy(S)$, $gini(S)$ or any other suitable measure.

It follows that the information requirements of decision tree learning algorithms are the same for both these splitting criteria; in both cases, we need the relative frequencies computed from the relevant instances. In fact, additional splitting criteria that correspond to other impurity measures can be used instead, provided that these measures can be computed based on the statistics that can be obtained from the data sets. Examples of such splitting criteria include misclassification rate, one-sided purity, one-sided extremes [6]. This turns out to be quite useful in practice since different criteria often provide different insights about data. Furthermore, as we show below, the information necessary for decision tree construction can be efficiently obtained from distributed data sets. This results in provably exact algorithms for decision tree induction from horizontally or vertically fragmented distributed data sets.

3.2 Distributed Information Extraction

Assume that given a partially constructed decision tree, we want to choose the best attribute for the next split. Let $a_j(\pi)$ denote the attribute at the jth node along a path π starting from the attribute $a_1(\pi)$ that corresponds to the root of the decision tree, leading up to the node in question $a_l(\pi)$ at depth l. Let $v(a_j(\pi))$ denote the value of the attribute $a_j(\pi)$, corresponding to the jth node along the path π. For adding a node below $a_l(\pi)$, the set of examples being considered satisfy the following constraints on values of attributes: $L(\pi) = [a_1(\pi) = v(a_1(\pi))] \wedge [a_2(\pi) = v(a_2(\pi))] \cdots [a_l(\pi) = v(a_l(\pi))]$ where $[a_j(\pi) = v(a_j(\pi))]$ denotes the fact that the value of the jth attribute along the path π is $v(a_j(\pi))$.

It follows from the preceding discussion that the information required for constructing decision trees are the counts of examples that satisfy specified constraints on the values of particular attributes. These counts have to be obtained once for each node that is added to the tree starting with the root node. If we can devise distributed information extraction operators for obtaining the necessary counts from distributed data sets, we can obtain exact distributed decision tree learning algorithms. Thus, the decision tree constructed from a given data set in the distributed setting is exactly the same as that obtained in the batch setting when using the same splitting criterion in both cases.

Horizontally Distributed Data

When the data is horizontally distributed, examples corresponding to a particular value of a particular attribute are scattered at different locations. In order to identify the best split of a particular node in a partially constructed tree, all the sites are visited and the counts corresponding to candidate splits of that node are accumulated. The learner uses these counts to find the attribute that yields the best split to further partition the set of examples at that node. Thus, given $L(\pi)$, in order to split the node corresponding to $a_l(\pi) = v(a_l(\pi))$, the information extraction component has to obtain the counts of examples that belong to each class for each possible value of each candidate attribute.

Let $|D|$ be the total number of examples in the distributed data set; $|A|$, the number of attributes; V the maximum number of possible values per attribute; n the number of sites; M the number of classes; and $size(T)$ the number of nodes in the decision tree. For each node in the decision tree T, the information extraction component has to scan the data at each site to calculate the corresponding counts. We have: $\sum_{i=1}^{n} |D_i| = |D|$. Therefore, in the case of serial distributed learning, the time complexity of the resulting algorithm is $|D||A| \cdot size(T)$. This can be further improved in the case of parallel distributed learning since each site can perform information extraction in parallel. For each node in the decisision tree T, each site has to transmit the counts based on its local data. These counts form a matrix of size $M|A|V$. Hence, the communication complexity (the total amount of information that is transmitted between sites) is given by $M|A||V|n \cdot size(T)$. It is worth noting that some of the bounds presented here can be further improved so that they depend on the height of the tree instead of the number of nodes in the tree by taking advantage of the sort of techniques that are introduced in [7, 8].

Vertically Distributed Data

In vertically distributed datasets, we assume that each example has a unique index associated with it. Subtuples of an example are distributed across different sites. However, correspondence between subtuples of a tuple can be established using the unique index. As before, given $L(\pi)$, in order to split the node corresponding to $a_l(\pi) = v(a_l(\pi))$, the information extraction component has to obtain the counts of examples that belong to each class for each possible value of each candidate attribute. Since each site has only a subset of the attributes, the set of indices corresponding to the examples that match the constraint $L(\pi)$ have to be transmitted to the sites. Using this information, each site can compute the relevant counts that correspond to the attributes that are stored at the site. The hypothesis generation component uses the counts from all the sites to select the attribute to further split the node corresponding to $a_l(\pi) = v(a_l(\pi))$. For each node in the decision tree T, each site has to compute the relevant counts of examples that satisfy $L(\pi)$ for the attributes stored at that site. The number of subtuples stored at each

site is $|D|$ and the number of attributes at each site is bounded by the total number of attributes $|A|$. In the case of serial distributed learning, time complexity is given by $|D||A|n \cdot size(T)$. This can be further improved in the case of parallel distributed learning since the various sites can perform information extraction in parallel. For each node in the tree T, we need to transmit to each site, the set of indices for the examples that satisfy corresponding constraint $L(\pi)$ and get back the relevant counts for the attributes that are stored at that site. The number of indices is bounded by $|D|$ and the number of counts is bounded by $M|A|V$. Hence, the communication complexity is given by $(|D| + M|A|V)n \cdot size(T)$. Again, it is possible to further improve some of these bounds so that they depend on the height of the tree instead of the number of nodes in the tree using techniques similar to those introduced in [7, 8].

Distributed versus Centralized Learning

Our approach to learning decision trees from distributed data based on a decomposition of the learning task into a distributed information extraction component and a hypothesis generation component sites provides an effective way to deal with scenarios in which the sites provide only statistical summaries of the data on demand and prohibit access to raw data. Even when it is possible to access the raw data, the distributed algorithm compares favorably with the corresponding centralized algorithm which needs access to the entire data set whenever its communication cost is less than the cost of collecting all of the data in a central location. It follows from the preceding analysis that in the case of horizontally fragmented data, the distributed algorithm has an advantage when $MVn \cdot size(T) \leq |D|$ since the cost of shipping the data is given by its actual size, which is given by $|D||A|$. In the case of vertically fragmented data, the corresponding conditions are given by $size(T) \leq |A|$ since the cost of shipping the data is given by its actual size, which has a lower bound of $|D||A|$. These conditions are often met in the case of large, high-dimensional data sets.

4 Data Integration in INDUS

The discussion of distributed learning in the preceding section assumed that it is possible to extract the information needed by the learning algorithm (e.g., counts of instances that satisfy specific constraints on the values of the attributes) from the distributed data sources. This is rather straightforward in the case of data sources that have a homogeneous structure and semantics. However the heterogeneity in the structure and semantics of data can make this task significantly more complex in real-world knowledge discovery tasks [2].

INDUS is designed to provide a unified query interface over a set of distributed, heterogeneous and autonomous data sources which enables us to view each data source *as if* it were a table. For the purpose of decision tree learning, we formulate queries whose executions return tables containing data of interest (e.g., counts). They are driven by the learning algoritm which used them whenever a new node needs to be added to the tree.

5 Summary and Discussion

Efficient learning algorithms with provable performance guarantees for knowledge acquisition from distributed data sets constitute a key element of any attempt to translate recent advances in our ability to gather and store large volumes of data into an ability to effectively use the data to advance our understanding of the respective domains (e.g., biological sciences, atmospheric sciences) and decision support tools. In this paper, we have precisely formulated a class of distributed learning problems and briefly presented a general strategy for transforming a class of traditional machine learning algorithms into distributed learning algorithms. We have demonstrated the application of this strategy to devise algorithms for decision tree induction (using a variety of splitting criteria) from distributed data. The resulting algorithms are *provably exact* in that the decision tree constructed from distributed data is identical to that obtained by the corresponding algorithm when it is used in the batch setting. This ensures that the entire body of theoretical (e.g., sample complexity, error bounds) and empirical results obtained in the batch setting carry over to the distributed setting. The proposed distributed decision tree induction algorithms have been implemented as part of INDUS, an agent-based system for data-driven knowledge acquisition from heterogeneous, distributed, autonomous data sources.

The distributed learning problem has begun to receive considerable attention in recent years. However, many of the algorithms proposed in the literature [9, 10, 11] do not guarantee generalization accuracies that are provably close to those obtainable in the centralized setting. Typically, they deal with only horizontally fragmented data. Furthermore, several of them are motivated by the desire to for scale up batch learning algorithms to work with large data sets by partitioning the data and parallelizing the algorithm. In this case, the algorithm typically starts with the entire data set in a central location; the data set is then distributed across multiple processors to take advantage of parallel processing. In contrast, in the distributed scenario discussed in this paper, the algorithm may be prohibited from accessing the raw data; even when it is possible to access the raw data, it may be infeasible to gather all of the data at a central location (because of the bandwidth and storage costs involved).

The algorithm proposed in [12], is closely related to our algorithm for learning decision trees from vertical fragmented data using entropy or infor-

mation gain as the splitting criterion. It provides a mechanism for obtaining counts from *implicit tuples* in the absence of a unique index for each tuple in the data set by simulating the effect of *join* operation on the sites without enumerating the tuples. In contrast, our algorithms assume the existence of a unique index, but are more general in other respects (ability to deal with both horizontal and vertical fragmentation, incorporation of multiple splitting criteria). Our approach can be modified using an approach similar to that used in [12] in the absence of unique indices.

Our work on the algorithms and software for data-driven scientific discovery has been driven, at least in part, by the needs of emerging data-rich disciplines such as biological sciences where there is an explosive growth in number, diversity, and volume of data being archived in publicly accessible, distributed, autonomous data repositories [2].

References

1. V. Honavar, L. Miller, J. Wong, Distributed knowledge networks, in: Proc. of the IEEE Conf. on IT, Syracuse, NY, 1998.
2. V. Honavar, A. Silvescu, J. Reinoso-Castillo, D. Caragea, C. Andorf, D. Dobbs, Ontology-driven information extraction and knowledge acquisition from heterogeneous, distributed biological data sources, in: Proceedings of the IJCAI-2001 Workshop on Knowledge Discovery from Heterogeneous, Distributed, Autonomous, Dynamic Data and Knowledge Sources, 2001.
3. N. Jennings, M. Wooldridge, Agent-oriented software engineering, in: J. Bradshaw (Ed.), Handbook of Agent Technology, AAAI/MIT Press, 2001.
4. R. Quinlan, Induction of decision trees, Machine Learning 1 (1986) 81–106.
5. L. Breiman, J. Friedman, R. Olshen, C. Stone, Classification and Regression Trees, Wadsworth, Pacific Grove, CA, 1984.
6. A. Buja, Y. Lee, Data Mining Criteria for Tree-Based Regression and Classification, 2000.
7. J. Shafer, R. Agrawal, M. Mehta, Sprint: A scalable parallel classifier for data mining, in: VLDB'96, Proc. of 22th Int. Conf. on VLDB, September 3-6, 1996, Mumbai (Bombay), India, Morgan Kaufmann, 1996.
8. J. Gehrke, V. Ganti, R. Ramakrishnan, W. Loh, Boat – optimistic decision tree construction, in: Proc. SIGMOD Conf., Philadelphia, Pennsylvania, 1999.
9. W. Davies, P. Edwards, Dagger: A new approach to combining multiple models learned from disjoint subsets, in: ML99, 1999.
10. P. Domingos, Knowledge acquisition from examples via multiple models, in: Proc. of the 14th ICML, Nashville, TN, 1997.
11. A. Prodromidis, P. Chan, S. Stolfo, Meta-learning in distributed data mining systems: Issues and approaches, in: H. Kargupta, P. Chan (Eds.), Advances of Distributed Data Mining, AAAI Press, 2000.
12. R. Bhatnagar, S. Srinivasan, Pattern discovery in distributed databases, in: Proc. of the AAAI-97 Conf., Providence, RI, 1997.