AN ABSTRACT OF THE THESIS OF

Stephanie Deutschman for the degree of Master of Science in Computer Science
presented on June 15, 2007.
Title: Accuracy Versus Cost in Distributed Data Mining.

Abstract approved:

Timothy A. Budd

A basic tradeoff to consider when designing a distributed data-mining framework is the need for a compromise between the cost of communication and computation resources and the accuracy of the mining results. This is essentially a decision of whether it is more efficient to communicate all of the data to a central site for analysis, possibly increasing the accuracy of the results, or is it more efficient to mine the data locally at each of the remote sites and then combine the results, possibly reducing the use of communication and computation resources. This research attempts the design, analysis, and implementation of an efficient distributed and cumulative learning algorithm with performance guarantees that are provable relative to its centralized or batch counterparts for knowledge acquisition from distributed data sources that will address this tradeoff.

This thesis also develops a methodical mathematical framework to describe this type of tradeoff, describes the reduction of the problem to a constrained optimization problem, and demonstrates techniques to balance cost and accuracy levels.

Accuracy Versus Cost in Distributed Data Mining

by
Stephanie Deutschman

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented June 15, 2007
Commencement June 2008

Master of Science thesis of Stephanie Deutschman presented on 15 June, 2007.

APPROVED:

_____

Major Professor, representing Computer Science

_____

Director of the School of Electrical Engineering and Computer Science

_____

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

_____

Stephanie Deutschman, Author

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

LIST OF TABLES

Accuracy versus cost in distributed data mining

Abstract

A basic tradeoff to consider when designing a distributed data-mining framework is the need for a compromise between the cost of communication and computation resources and the accuracy of the mining results. Is it more efficient to communicate all of the data to a central site for analysis, possibly increasing the accuracy of the results, or is it more efficient to mine the data locally at each of the remote sites and then combine the results, possibly reducing the use of communication and computation resources? This research attempts the design, analysis, and implementation of an efficient distributed and cumulative learning algorithm with performance guarantees that are provable relative to its centralized or batch counterparts for knowledge acquisition from distributed data sources that will address this tradeoff.

This thesis also develops a methodical mathematical framework to describe this type of tradeoff, describes the reduction of the problem to a constrained optimization problem, and demonstrates techniques to balance cost and accuracy levels.

1 Combination Strategies

1.1 Introduction

A learning algorithm frequently is applied to a single set of data to construct an evaluation model, regardless of the size of that data set. A single predictive model is built to discover the underlying data distribution properties and to predict the class value of uncharacterized data instances. For organizational and technical reasons, applications to analyze data on high-performance distributed computing frameworks are shifting away from this sort of centralized computing model. One reason for this shift is that centralization is difficult due to the requirement of having to communicate multi-terabyte data sets over very long distances. A second reason involves the fact that data centralization may violate privacy legislation, expose business secrets, or pose other challenges to organizational policy. Examples of these challenges appear where the relevant data distribution includes multiple parties: government bodies such as the US Food and Drug Administration, commercial organizations including drug companies or hospitals, and non-governmental organizations such as public-health organizations and charities (Kantardzic and J. Zurada, 2005, p. 8). Every organization is restricted by regulations such as privacy legislation or corporate requirements affecting proprietary information that could give competitors a commercial advantage. The shift away from centralized data storage produces several, often geographically distributed, data sets that may come from multiple data distributions, and prompts

the development of new techniques for manipulating data before applying a learning algorithm.

In one common data manipulation method, the complete data set is communicated to a single centralized site, so that the standard methods of applying a learning method and building a single predictive model may be applied. Centralized mining is a naïve strategy that is effective only for very small datasets. This method of distributed mining usually produces the most accurate predictive models; however, for several reasons, it may be a more expensive procedure. These reasons include

1. Large increases in network traffic as the data is transferred
2. Problems that may arise related to data delivery, cleaning, and aggregation
3. The data processing algorithm complexity, in itself, may become a problem in cases when the algorithm complexity is quadratic and is used for a distribution of k local sites that each contains n data points. In this situation, $O(kn^2)$ is necessary to apply the algorithm separately to k individual subsets of the data, but $O(k^2n^2)$ is required if the data is combined as a single set holding kn data points.

A second technique often used in distributed mining is to develop a separate predictive model at each distinct local data site and then combine the results of these models. This localized mining technique avoids the time-consuming process of moving large datasets across corporate networks and, due to the ubiquity of Web-services based architectures and the tremendously large amount of data available for mining, is the cheapest and fastest, but is also the least accurate.

There are several combinations of these processes in which some data is left in place and mined locally while some data is moved, and then the subsequent data

models are combined at a central location. Due to the ever-increasing reliance on high-speed communications networks, these combination methods are becoming significantly more practical and demonstrate a balance between the level of communication cost and the accuracy of the model results. Table 1 shows this balancing effect.

Table 1.1: The trade-off between cost and accuracy with different distributed data mining strategies.

Data Mining Strategies	Cost	Accuracy
In-place	Low	Low
Combined	Balanced	Balanced
Centralized	High	High

An increasing interest in, and desire for, distributed mining processes that are efficient combinations of the in-place and centralized strategies outlined above drives this study. We detail our examination of combinational mining processes in a situation that uses a high performance network infrastructure and a cost function that represents both the computational and communication costs in the text that follows. Our research will show that, in this context, the problem of finding the most beneficial tradeoff between efficiency and accuracy can be reduced to a convex programming problem that minimizes a cost function subject to a given error restriction.

1.2 Background and related work

As outlined above, there are two common approaches to distributed data mining, centralized learning and local learning. Centralized learning moves all the data to a central site for analysis and creation of predictive models while local learning builds predictive models locally at each site and then moves the models to a central location where they are combined.

Ensemble learning is a common technique used to combine the models created at geographically distributed sites. Methods for combining models in an ensemble include meta-learning (Stolfo, et. al., 1997), knowledge selection (Guo, et. al., 1997), voting schemata (Ditterich, 1997), model selection (Raferty, et. al., 1996), stacking, mixture of experts, and Bayesian model averaging.

Many systems designed for distributed data analysis have been developed, and each of the systems implements different types of learning algorithms. The Kensington system, developed by Guo et al (Guo, et. al., 1997) utilizes knowledge probing that examines learning with a black box perspective and builds a predictive model by inspecting both the input and output of each local model, coupled with the desired output. Kargupta, et. al. developed the BODHI system (Kargupta, et. al., 1999) to employ collective mining methods that rely on techniques from Fourier analysis when combining the individual models into an ensemble. Grossman, et. al. is finalizing the creation of a distributed data mining system known as Papyrus (Grossman, et. al., 2000). Papyrus is intended to support

different model and data schemes, including local learning, centralized learning, and various combination strategies, in other words, hybrid learning. Future development planned for the Papyrus system includes work to develop a method to choose an information transfer strategy that can be optimized for a specific data mining task.

An assortment of load balancing methods has been applied to parallel computing techniques over the years. Load balancing is intended to find an optimal strategy for transferring data to the various nodes of a supercomputer or to network workstations. Cheung describes a method of load balancing that optimizes the communication efficiency of parallel computing over a network of clustered compute nodes. Other examples of load balancing have been discussed, such as load balancing in a heterogeneous computing environment (Grimshaw et. al., 1993) and in distributed object computing systems (Zaki et. al., 1997), but these load balancing techniques do not address issues, such as how to combine predictive models or how to ensure the accuracy level of the subsequent predictive system, that are important in the field of distributed data mining.

An important, though often overlooked, data mining interest is research into the learning curves that show the relationship between the size of the initial data set and the accuracy of the predictive model(s) that are built on the data. Learning curves are used to evaluate how a learning scheme's predictive performance responds to increasing amounts of training data. Cortes states that, in most cases, exposing a model to more data reduces the predictive error, although not to zero

(Cortes et. al., 2001). Learning curves change in shape depending on many factors including the data quality and the type of models, but they also share several common features that may be used when deciding on the tradeoff between efficiency and accuracy in the data mining process.

1.3 Computational model

We completed our research using a computational model consisting of processors separated by hundreds of miles, one in Palo Alto, California, one in Houston, Texas, and one in Corvallis, Oregon. Each of the processors is a Hewlett-Packard C160 running HP-UX v.11. We measured the roundtrip communication latency using the *ping* command and the steady transfer rate between any two machines was measured using FTP; the average measurements are displayed in Tables 2 and 3.

Table 1.2: Average roundtrip communication latency (ms)

	Corvallis	**Houston**	**Palo Alto**
Corvallis	n/a	105	90
Houston	105	n/a	70
Palo Alto	90	70	n/a

Table 1.3: Steady transfer rate between processors (kbps)

	Corvallis	**Houston**	**Palo Alto**
Corvallis	n/a	280	235
Houston	280	n/a	265
Palo Alto	235	265	n/a

We repeated these experiments on several days at different times of the day to determine an average performance level; communication latency times varied only a little, the steady transfer rate varied slightly more, but neither variance was statistically significant.

For the purposes of this research we kept the problem formulation very simple; however in actual data mining processes there are frequently many more than three processing sites, the various processors may run at differing speeds, and the data partitions are not of equal sizes. The framework developed in this paper is capable of handling each of these possibilities while computing an optimal combination strategy.

1.3.1 Network configuration

The network used in our research was formally expected to include n different sites. The cost of processing data, in dollars per record, at the i^{th} node into a predictive model is considered to be v_i and the optimal cost of moving data from the i^{th} to the j^{th} node via the cheapest route is μ_{ij}. One of the nodes is assigned the role of network root and is where the overall combined results are calculated.

1.3.2 Building Predictive Models

A common assumption in distributed data mining is that a choice is required at each network node: move the raw data across the network to another node for processing, or process the data locally, create a predictive model, and then convey the model across the network for possible continued processing. The data, or some segment of the data, also might be both manipulated locally and shared with another processing node. This outlook lays a foundation for model building that involves the following steps:

1. Re-distribute raw data across the network

2. Compute a predictive model at every node
3. Transfer all local predictive models to the root node
4. Combine all models into one predictive model at the root node.

Let the R^{th} node be selected as a network root, where $1 \leq R \leq n$. Let D_i represent the amount of data initially stored at the i^{th} node. Following the re-apportioning of data completed in Step 1, the i^{th} node contains $\tilde{D}_i$ collected data. Let M_i be the size of the predictive model developed in Step 2 from $\tilde{D}_i$, which will later be transferred to the network root node. We expect that when data is processed into a predictive model the amount of data is compressed equally for each node with a coefficient α, so that

$$M_i = \alpha \tilde{D}_j \qquad (1.1)$$

The objective of this exercise is to establish the series of data transfers across the network that minimizes the total cost of building predictive models. In general, the approach outlined above will work with more complicated data movement and model creation strategies. As an example, a model building procedure could first move some or all of the data, then clean the data, produce predictive models, move the models, and ultimately combine the models. In this situation the cost function, strategy, and error function would be more complicated, but of the same generic form as shown above in Equation 1.1.

1.4 A method for finding the most favorable strategy for data and model partitioning

1.4.1 Strategies

A strategy X is defined as a matrix of numbers

$$X = \left[x_{ij} \right]_{i,\, j=1}^{n} \qquad (1.2)$$

in which x_{ij} represents the portion of data D_i that is transferred from the i^{th} node to the j^{th} node for processing. The segment x_{ij} may represent the percentage of the data to be shared between nodes rather than the amount of data; however, if the initial data distribution is known the two methods are analogous. The amount of data, x_{ij}, is added to $\tilde{D}_j$, the amount of data in node j after redistribution of data across the network; processed into a predictive model; and eventually delivered to the network root node where it is incorporated as a part of M_j. Calculating a strategy is limited to a situation where

$$0 \le x_{ij} \le D_i \qquad (1.3)$$

and the layout of the network may impose additional constraints such as fault tolerance, network speed, and traffic restrictions.

1.4.2 Local and Centralized Strategies

A centralized strategy is a strategy $X_0 = X_0(R)$ in which the data from all nodes is moved, in the first step of model building as outlined above, to the root node R for further processing. In this type of strategy $x_{iR} = D_i$ for each i, and the

remaining x_{ij} entries in the strategy matrix are zero. Assigning different nodes as the root node leads to the creation of up to n different centralized strategies.

A local strategy is defined as a strategy X_1 where all data is processed locally. A local strategy may be considered as fully distributed because all data processing is completed at distributed sites. This strategy is the equivalent of $x_{ii} = D_i$ for each i with each of the x_{ij} entries in the strategy matrix equal to zero.

Combination strategies demonstrate a compromise between fully centralized strategies X_0 and fully localized or distributed strategies X_1.

1.4.3 The Cost Function

Finding the most effective strategy for building predictive models using the method outlined above is easily reduced to a constrained optimization problem. We compute the overall cost function for a strategy X as

$$C(X) = \sum_{ij} \left(\mu_{ij} x_{ij} + \nu_j x_{ij} + \mu_{jR} \alpha x_{ij} \right) = \sum_{ij} c_{ij} x_{ij} \qquad (1.4)$$

In this equation, the first term expresses the cost of moving data; the second term is a calculation of the cost to construct predictive models from the data, and the final term represents the cost of transferring the predictive models to the root node for aggregation. For convenience, we define the coefficients as

$$c_{ij} = \mu_{ij} + \nu_j + \alpha \mu_{jR} \qquad (1.5)$$

The coefficient μ_{ij} represents the cost of network communication per unit volume of data between nodes *i* and *j*, and the coefficient ν_j corresponds to the cost per unit volume of data for an algorithm to process data and build a statistical model at node *j*. We establish *R* as the network root node, so μ_{jR} is the cost of moving data from node *j* to the root. We do not represent the term that expresses the cost of the final step to combine the results at the network root in this cost function because, regardless of the strategy *X*, the same amount of results must be processed at the root node. This is shown in the following equation

$$\sum_j M_j = \sum_j \alpha \tilde{D}_j = \alpha D \qquad (1.6)$$

where *D* is the full initial amount of the entire data set and α is the compression coefficient as shown in Equation 1.1 above. This demonstrates that the term in question is a constant and may be removed without any loss of generality. We may also discount the cost of combining the results at the network root when the cost of combining models is insignificant when compared to the remaining cost factors. Overall, combining several predictive models into a voting ensemble is much less costly than building the predictive models or transferring data across a network.

Our research first assumes a linear cost of data processing, which leads to a linear function optimization problem. Algorithms that are more complex generally propagate non-linear cost functions, many of which are convex, and may be solved using this same approach. The actual values of various coefficients are estimated

depending on factors such as the business infrastructure, the specific algorithms used, and the network throughput levels, among others.

The cost is different for each particular centralized strategy. Let C_0 be the best cost achievable using a centralized strategy, and let $C_1 = C(X_1)$ represent the cost of a localized strategy with the same network and data parameters. The localized strategy may not always be the least expensive option. In a situation where the data mining algorithm demands a high level of resources in the data processing stage, perhaps because of a large algorithmic complexity, the cost of exercising these resources may both be significant compared to the communication costs and vary considerably from node to node. Situations like this may make it more cost-effective to move the data to a site where processing is cheaper rather than calculating a model locally. In any situation, the most cost effective policy is determined by minimizing the cost function.

1.4.4 Error

Two factors introduce error when creating predictive data models: the characteristics of the data and computational algorithm, and the method of processing the data at distributed sites. The first introduction of error occurs because the characteristics of the data and algorithm may lead to reduced accuracy in the results despite the choice of strategy. This error term is designated ε_0 and is commonly used in statistical modeling. Calculation of this type of error may be

accomplished using a validation set but is beyond the scope of our research and is examined in standard books on applied statistics.

Accuracy also is lost when data is processed locally at multiple distinct nodes in lieu of moving the data to a root node and processing it there. Any procedure used to develop a statistical model for distributed data also may be used to create a model when all of the data is stored at a single location. It is important to note, however, that there are some tactics that may be used when the data is centralized that are not available for use with distributed data sets, and this may lead to the creation of inferior data models for these distributed data. Ensemble learning is a typical case of this situation because although several models may be trained the results are often more valuable if the individual learning models are built from a centralized data set.

As an example, assume an instance in which the data is heterogeneous and the distributed sites each maintain a database containing a different type of information. Processing each node's data locally will create a predictive model that is useful only when classifying data that is of the same type as the data on which the model was built. Using a collection of these locally built models as a voting ensemble will yield inaccurate results whenever the specific type of a new data record is unknown since most of the individual models will be unacceptable predictors.

This example is typical of real-world distributed systems where many huge data sets are partitioned into subsets of a more manageable size in a manner that

makes them heterogeneous. Common ways of splitting a data set include storing the data sorted by one or more attributes, or splitting the data sequentially into smaller pieces. If, for example, a cell phone database was earlier sorted by the number of minutes used by a customer, different subsets would hold data on customers in different minute ranges, but this difference can be overlooked during further processing which may lead to the creation of biased local models.

A separate learning curve is established when learning the data from each subset of the data for an individual model. Murata studies the shapes of these learning curves and indicates that the overall error level of an ensemble of models can be reduced by sharing portions of data between processing sites (Murata et. al., 2001). The error function E(X) is a combination of one learning curve for each dimension in the X-space of strategies, in other words, a superposition of the learning curves representing each variable on which the strategy based its learning. The surface of the error function E(X) may be thought of as a surface constructed from collections of learning curves positioned parallel to each of the x_{ij}-axes in the X-space. To learn the first data subset by the second model, where the data from other subsets has been learned to the same degree demonstrated by the other components of the strategy, a single component x_{12}, of strategy X, is altered and settings are assigned for each of the remaining components of X.

The level contours of the error surface E(X), specifically the contours $\{X : E(X) = const\}$, produce a group of related convex sets within the space of

possible strategies. Figure 1 displays this, and this observation is important in the process of determining the optimal data allocation strategies.

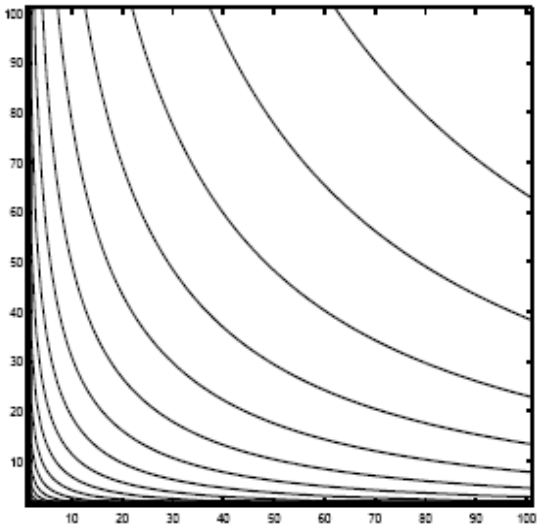

Figure 1: A 2-dimensional slice of the level contours for an error function obtained from mining data in the UCI KDD Archive Forest Cover Type dataset.

The coordinate axes of Figure 1 represent the percentages of data coming from two distinct sites. The error value is greatest for a localized mining strategy, (0%, 0%), and lowest in a centralized strategy (100%, 100%). The cost function commonly decreases in the reverse direction, presenting the likelihood of an optimal combination strategy.

It also may be demonstrated that, if all distributed data is from a single data distribution, the error function will be constant on the linear sets $\{X : \Sigma_i x_{ij} = \gamma_j\}$, where γ_j are constants, because the only factor that reduces the error is the amount of data processed, $\tilde{D}_j = \Sigma_i x_{ij}$, and not the source of the data. In this situation, the level contours of the error function will be linear. Murata indicates (Murata, et. al., 2003) that when the shape of the error curves is of the type $\xi_0 + \xi_1/x$, where x is the

amount of training data and ξ_1 and ξ_2 are known constants, the error rate of a particular model *j* is of the type

$$\begin{aligned} E_j &= \xi_0 + \frac{\xi_1}{\tilde{D}_j} \\ &= \xi_0 + \frac{\xi_1}{\sum_i x_{ij}} \end{aligned} \qquad (1.7)$$

The experiments we performed to support our research indicate that if the data is heterogeneous the level contours in the error function will show a significantly more prominent convexity. This heightened convexity is caused by a faster decrease in the error along the paths in X-space that represent a blend of data from multiple sources, and a slower decrease accompanying paths that are parallel to coordinate axes.

We write an error function for model *j* as

$$E_j(X) = b_{0j} + \frac{1}{b_{j0} + \sum_{i=1}^{n} b_{ij}\, x_{ij}^{p_j}} \qquad (1.8)$$

where *n* is the number of data subsets, b_{ij} are estimated parameters, and the exponent $p_j \in (0, 1]$ specifies the degree of convexity. The overall error function E(*X*), or the upper bound for E(*X*), is then determined subject to how the local models are combined. For example, if the models are combined in an averaging ensemble the inconsistency among the individual models always results in the error

of the ensemble being lower than the average error of the distinct local models. These circumstances lead to the error bound

$$E(X) \leq \frac{1}{k}\sum_{j} E_j(X) \qquad (1.9)$$

Other ensemble learning methods, such as boosting and bagging (Dietterich, 2000), may be used to improve the accuracy level of the combined model, but these methods may also cause the error function to be of a much more complex mathematical structure.

Many other mathematical forms of the error function E(X) that satisfy the convexity property are possible and may be used without significant changes to the model building algorithm. The two most sensible ways to define an error function are to either

1. Depend on previous knowledge of a specific type of data mining problems
2. Use a set of sample data in a simulation that will establish the parameters of the error function.

Our research will show an example of this methodology in the next section.

Once we identify the parameters of the error function, the minimum and maximum values ε_0 and ε_1 of the function may be approximated in order to find the range

$$\varepsilon_0 \leq E(X) \leq \varepsilon_1 \qquad (1.10)$$

With this model, it is clear that the model drops to its minimum error value ε_0 when all of the data is present at a single processing site as occurs in centralized

strategies X_0, and reaches its maximum error value ε_1 when the data is processed with an in-place strategy X_1 that allows no data sharing.

The data mining error is a random variable that is different for distinct experiments. The method based on learning curves, as described above, focuses on the average error for a specific category of experiments, or on the upper bound of the corresponding error, but not necessarily on the actual error rate for any specific experiment or instance of a type of experiment.

1.4.5 Optimization of the Strategy

The next endeavor of our research is to find a strategy $X = [x_{ij}]$ that is a solution to the following optimization problem where ε_{max} is the maximum allowable error level, the vector D is given by the initial data distribution, and c_{ij} is defined by Equation 1.5.

$$\begin{cases} \min_{[x_{ij}]} C(X) = \sum_{ij} c_{ij} x_{ij} \\ 0 \le \sum_{j} x_{ij} \le D_i \\ E(X) \le \varepsilon_{\max} \end{cases} \qquad (1.11)$$

The optimal solution to this set of equations is the strategy X^* that represents the lowest cost level $C^* = C(X^*)$ demonstrated among all strategies that exhibit acceptable accuracy levels.

The first two equations in the optimization problem represented in Equation 1.11 define a linear programming problem in domain B, which is a convex polyhedron with a multi-dimensional rectangle boundary in X-space. The solution

to this linear programming problem, $X_{\Diamond}$, is easy to calculate and represents the best cost possible when no accuracy restrictions are imposed (Chatval, 1983). Graphically, $X_{\Diamond}$ corresponds to the lowest vertex of the polytope B, in which the direction is established by the level of the cost function. As noted previously, in some cases $X_{\Diamond}$ will not be the same as the localized strategy; however, if $X_{\Diamond}$ meets the required accuracy levels then the optimal strategy $X^* = X_{\Diamond}$; if the accuracy level is not adequate the optimization problem of Equation 1.11 must be re-solved to meet all three of the required conditions.

The third equation of the optimization problem is another constraint, and it is a convex constraint since E(X) has context level contours. Because of this characteristic, the collection $\{B_\varepsilon\}$ is a growing family of convex sets within polytope B where the sets $B_\varepsilon = \{X \in B : \mathrm{E}(X) \leq \varepsilon\}$ are defined with an accuracy threshold parameter $\varepsilon \in [\varepsilon_0, \varepsilon_1]$. Finding an optimal combined strategy may now be reduced to minimization of a linear function E(X) on a convex set $B_{\varepsilon_{\max}}$, which is a thoroughly researched class of optimization problem that is easily solved using standard techniques (Lewis and Borwein, 2000).

Our supplementary investigation into the level contours displayed by E(X) did not reveal any indication that it is possible to reverse the direction of convexity of the contours $\{X : \mathrm{E}(X) = const\}$. A convexity reversal of this sort could occur if, for example, $p_j > 1$ in the error function displayed in Equation 1.8, or in functions with comparable shape. If this type of reversal were to occur, the optimization

problem outlined above would need to be modified in the following way: the sets $A_\varepsilon = \{X \in B : E(X) \le \varepsilon\}$ will now be convex and the sets $\{B_\varepsilon\}$, as established earlier, will be the complements of $\{A_\varepsilon\}$. An example of this situation is any function, such as $1/(x^2 + y^2)$, whose level contours are circles centered on the origin. When the error level for the modified problem is set to the maximum level, ε_{max}, the optimal solution X^* still corresponds to the lowest point of the convex set $B_{\varepsilon_{max}}$; however, it may be shown using the convexity argument that this lowest point must be at the intersection of the error level surface $\{X : E(X) = \varepsilon_{max}\}$ and one of the edges of the domain polytope.

This modification to the cost optimization problem forces a change in the procedure to find X^*. The adjusted method first uses linear programming techniques to find the lowest point $X_\diamond$ of B, and then moves along each edge in the direction of increasing cost until the edge intersects the error level surface $E(X) = \varepsilon_{max}$. This intersection point occurs when travel along the polytope edge reaches the lowest boundary of the set of acceptable strategies $B_{\varepsilon_{max}}$. The lowest of this set of intersection points is X^*. If domain B is a simple multidimensional box, such as when the linear restrictions are defined as in Equation 1.11, the reduced number of edges and intersection points simplifies the geometry of this modified procedure.

This method of determining the optimal strategy for compromise between localized and centralized data mining procedures is both straightforward and

functional; however, it does not address the process of selecting which instances of the data to move, only the fraction or percentage of the whole to move. In the most primitive case, we assume that the local data sets are uniform and that random samples of the local data will be moved.

1.5 Case Study: Forest Cover Type

We tested our research model with experiments involving a small number of datasets from the UCI Machine Learning Repository and the UCI KDD Archive. Our experimental results were similar for all of the datasets tested, but some datasets showed almost no increase in accuracy when using centralized processing compared to localized processing. The datasets that exhibited little change in accuracy levels also displayed a high classification error, even under the most favorable conditions of centralized learning processes. It appears, after further investigation of the datasets, that the scant increase in accuracy levels may be due to the high level of homogeneity and high intrinsic noise levels in the data.

The following example will demonstrate how we used the method developed in this investigation to generate a non-trivial optimal cost solution. This example is based on the Forest Cover Type dataset from the UCI KDD Archive. The dataset contains 581,012 data instances, each with 54 attributes grouped in 12 separate measures.

1.5.1 Data Preparation

For this example we use $n = 3$ distinct sites that contain distributed, possibly heterogeneous data. The concern of our research is to determine how much data should be transferred among sites prior to beginning the data processing activities while still maintaining acceptable limits on the accuracy of the results. The models developed during processing are collected at a single node, the root node $R = 1$, and

combined into an ensemble. To represent these conditions we split the Forest Cover Type dataset into three equally sized portions, that each represents one of the individual sites. We also set aside a portion of the data to use as a validation set. Because the data stored in the UCI KDD Archive was sorted, a common practice, the three subsets created during this sequential partitioning are heterogeneous. We use a C4.5 classification tree, an efficient decision tree algorithm (Ruggieri, 2002), as our decision model in this example. We built the decision tree models using a student version of GhostMiner®, a data mining and decision support system produced by Fujitsu. We combined the models into a voting ensemble such that, for each new data instance, each of the models will make its individual prediction and then a majority vote is taken to determine the final prediction of the ensemble.

Each component x_{ij} of a strategy indicates the percentage of data that is shared between sites i and j, and all data stored locally will be used, along with sections of data introduced by other models, for building local models. For this reason, $x_{ii} = 1$ for all nodes i. This definition of the elements in this data transfer problem results in a strategy matrix

$$X = \begin{pmatrix} 1 & x_{12} & x_{13} \\ x_{21} & 1 & x_{23} \\ x_{31} & x_{32} & 1 \end{pmatrix}, \; x_{ij} \in [0, 1] \qquad (1.12)$$

Our task now is to determine values for the remaining six components of the matrix.

1.5.2 Estimation of the Error Function

We considered each local model separately when estimating the error introduced in building C4.5 trees on a specific distributed collection of data. As an example, we developed Model 1 using the entirety of data subset D_1, an x_{21} portion of subset D_2, and an x_{31} portion of subset D_3. Using an error function of the same type as Equation 1.8, explained above, the error demonstrated by Model 1 is, in effect, a function of two variables, x_{21} and x_{31}, and is shown in the following equation

$$E_1(X) = b_{10} + \frac{1}{b_{01} + b_{21}x_{21}^{p_1} + b_{31}x_{31}^{p_1}} \qquad (1.13)$$

Because $x_{11} = 1$, $b_{01} + b_{11}x_{11}^{p_1}$ is combined in the constant b_{01} and written as such in the first term of the denominator. To calculate the coefficients, we arranged the values of $E_1(X)$ on the square $(x_{21}, x_{31}) \in [0, 1]^2$ by shifting portions of data, indicated by a pair (x_{21}, x_{31}), from nodes 2 and 3 to node 1; building a C4.5 tree model; and computing the model's accuracy by applying it to a validation set. We averaged the results of a five-fold cross validation of the training data, and these tabulated error values are shown in Table 4 through Table 6 in which the rows and columns correlate to the individual x_{ij} matrix elements for values such that $0 \le x_{21} \le 1$ and $0 \le x_{31} \le 1$.

Table 1.4: Misclassification Error Rate (%) for Model 1

x_{21} \ x_{31}	**0**	**.25**	**.50**	**.75**	**1**
0	15.0	11.1	11.8	11.5	11.1
.25	11.6	5.2	4.4	4.2	3.9
.05	11.3	4.9	4.2	3.9	3.6
.75	10.9	4.5	3.7	3.6	3.2
1	10.4	4.4	3.6	3.2	2.9

Table 1.5: Misclassification Error Rate (%) for Model 2

x_{21} \ x_{31}	**0**	**.25**	**.50**	**.75**	**1**
0	11.7	8.3	7.6	7.2	7.5
.25	9.3	5.4	4.6	4.4	4.0
.05	10.4	4.5	3.8	3.6	3.3
.75	10.3	4.4	3.7	3.4	3.1
1	10.4	4.3	3.6	3.2	2.9

Table 1.6: Misclassification Error Rate (%) for Model 3

x_{21} \ x_{31}	**0**	**.25**	**.50**	**.75**	**1**
0	17.5	12.5	9.0	8.6	7.5
.25	12.5	4.9	4.6	4.3	4.0
.05	11.5	4.3	3.9	3.6	3.3
.75	11.3	4.0	3.7	3.4	3.1
1	11.1	3.9	3.6	3.2	2.9

The error functions for these three models each display the expected behavior of attaining the minimum error value when all the data is collected at a single processing site and reaching the maximum error value when no data is shared. The data presented in the above tables makes it easy to appreciate that a purely localized mining strategy is liable to be of inferior quality and that sharing data is important, as even a small amount of sharing generates a sharp improvement in accuracy levels. Sharing small amounts of data is relatively cost effective,

leading to anticipation that combination strategies will exhibit a good balance between the communication and computing costs and the accuracy of the results.

The values obtained from these calculations are then used to fit the parameters of the error functions (as shown in Equation 1.8). We used a student version of the MATLAB package to calculate least-squares estimates for the parameters and the resulting error function formulas are:

$$E_1(X) = -.4812 + \frac{1}{1.1347 + .1674(x_{21})^{.1162} + .1716(x_{31})^{.1162}}$$
$$E_2(X) = -.4229 + \frac{1}{1.3626 + .0925(x_{12})^{.1254} + .1974(x_{32})^{.1254}} \qquad (1.14)$$
$$E_3(X) = -.3769 + \frac{1}{1.3326 + .2324(x_{13})^{.1494} + .3076(x_{23})^{.1494}}$$

1.5.3 Optimization

When both the error function and the cost function are established, the optimization problem detailed in Equation 1.11 may be solved by using standard techniques such as Newton's Method, the Simplex Method, or the Kuhn-Tucker theorem. A trivial change to the example highlighted in our research makes it possible to arrive at an analytical solution to the optimization problem in Equation 1.11. This modification is the addition of the requirement that each local model satisfy the property $E_j(X) \leq \varepsilon_{max}$ where $j = 1$, 2, and 3. This small adaptation permits the division of Equation 1.11 into three smaller optimization problems:

$$\begin{cases} \min_{[x_{ij}]} C(X) = c_{21}x_{21} + c_{31}x_{31} \\ 0 \le x_{ij} \le 1 \\ E_1(X) = b_{10} + \dfrac{1}{b_{01} + b_{21}x_{21}^{p_1} + b_{31}x_{31}^{p_1}} \le \varepsilon_{\max} \end{cases} \tag{1.15}$$

$$\begin{cases} \min_{[x_{ij}]} C(X) = c_{12}x_{12} + c_{32}x_{32} \\ 0 \le x_{ij} \le 1 \\ E_2(X) = b_{20} + \dfrac{1}{b_{02} + b_{12}x_{12}^{p_2} + b_{32}x_{32}^{p_2}} \le \varepsilon_{\max} \end{cases} \tag{1.16}$$

$$\begin{cases} \min_{[x_{ij}]} C(X) = c_{13}x_{13} + c_{23}x_{23} \\ 0 \le x_{ij} \le 1 \\ E_3(X) = b_{30} + \dfrac{1}{b_{03} + b_{13}x_{13}^{p_3} + b_{23}x_{23}^{p_3}} \le \varepsilon_{\max} \end{cases} \tag{1.17}$$

Temporarily disregarding the $0 \le x_{ij} \le 1$ constraint and using standard optimization techniques and some mathematical transformations it can be established that an optimization problem

$$\begin{cases} \min_{[x]} C(X) = c_{ij}x_{ij} + c_{kj}x_{kj} \\ E_j(X) = b_{j0} + \dfrac{1}{b_{0j} + b_{ij}x_{ij}^{p_j} + b_{kj}x_{kj}^{p_j}} \le \varepsilon_{\max} \end{cases} \tag{1.18}$$

has an analytical solution

$$x_{ij} = \left(b_{ij}c_{kj}\right)^{\frac{1}{1-p_j}} \left[\frac{\frac{1}{e_{\max} - b_{j0}} - b_{0j}}{\left(b_{ij}c_{kj}^{p_j}\right)^{\frac{1}{1-p_j}} + \left(b_{kj}c_{ij}^{p_j}\right)^{\frac{1}{1-p_j}}} \right] \qquad (1.19)$$

An optimal solution that also satisfies the constraint $0 \leq x_{ij} \leq 1$ is then either the result of Equation 1.19, or one of the easily detected intersection points between the error curve $E_j(X) = \varepsilon_{\max}$ and the perimeter of the square $[0, 1]^2$.

1.5.4 Cost

The error function for a predictive model is traditionally determined by the data processing algorithms in use and the quality of the underlying data, while the cost function is built on considerations such as the hardware systems, network infrastructure, and software systems that are in place. For this research, we allowed a high level of discretion in the choice of cost function.

It is necessary that we represent the cost of data processing by a matrix of v_{ij} values instead of a vector v_j in any situation where the x_{ij} components of a strategy express the percentages of data, rather than the actual amounts of data, $x_{ij}D_i$, that are shared. In this case, because we initially partitioned the data in this study such that there were equal amounts of data at each local site, it is not required to represent the cost as a matrix and it is acceptable to represent it as a vector (v_1, v_2, v_3). The communication cost between each two nodes is defined as uniform, so only knowledge of the values (μ_{12}, μ_{13}, μ_{23}) is required.

We accept the cost of shipping the five decision tree models that are created at local nodes to the root node to be combined into an ensemble as inconsequential because the size of a decision tree model is significantly smaller than that of the data from which it was created. If this assumption is not made, there are superficial modifications that must be made to the following procedures; however, as we observed above, regardless of the strategy that is chosen, collecting the three models at the root node will have the same cost and so will not effect the optimization. For this reason, $\alpha = 0$ (in Equation 1.1) and

$$\left[c_{ij}\right]=\left[\mu_{ij}+v_j+\alpha\mu_{j1}\right]=\begin{bmatrix} v_1 & \mu_{12}+v_2 & \mu_{13}+v_3 \\ \mu_{12}+v_1 & v_2 & \mu_{23}+v_3 \\ \mu_{13}+v_1 & \mu_{23}+v_2 & v_3 \end{bmatrix} \qquad (1.20)$$

1.5.5 Localized and Centralized Strategies

There are three centralized strategies, depending on which node is selected to be the root node, for moving all the data to a single node and building one model:

$$X_0(1)=\begin{bmatrix}1&0&0\\1&0&0\\1&0&0\end{bmatrix} \quad X_0(2)=\begin{bmatrix}0&1&0\\0&1&0\\0&1&0\end{bmatrix} \quad X_0(3)=\begin{bmatrix}0&0&1\\0&0&1\\0&0&1\end{bmatrix} \qquad (1.21)$$

The most desirable centralized cost is the smallest of the three values calculated from the equations:

$$\begin{aligned} C\left(X_0(1)\right) &= \mu_{12} + \mu_{13} + 3v_1 \\ C\left(X_0(2)\right) &= \mu_{12} + \mu_{23} + 3v_2 \\ C\left(X_0(3)\right) &= \mu_{13} + \mu_{23} + 3v_3 \end{aligned} \qquad (1.22)$$

and the lowest centralized error value for $E(X_0) = \varepsilon_0 = 3.4\%$, as shown above in Table 2.

It may be possible to further improve the accuracy in this centralized strategy by building an ensemble of models at the site to which all the data was moved, but this may add extra costs due to increasing the complexity of the data processing portion of the strategy.

The localized strategy, in which there is no data sharing, is represented by the matrix:

$$X_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (1.23)$$

with a cost $C_1 = v_1 + v_2 + v_3$ and high error levels, as shown in previous tables:

$$\begin{cases} E_1\left(X_1\right) = 15.0\% \\ E_2\left(X_1\right) = 11.7\% \\ E_3\left(X_1\right) = 17.5\% \end{cases} \qquad (1.24)$$

This demonstrates that the localized strategy is not acceptable due to the poor accuracy levels. The centralized strategy also may be unacceptable because, depending on the relative values of the communication and data processing costs, it

may be too expensive, forcing lower accuracy levels in trade for a distinct reduction in cost.

1.5.6 Optimal Solution

In order to examine the results of using different combinations of μ_{ij} and v_j in the mining solution, we set the accuracy threshold at $\varepsilon_{max} = 8\%$. The values defined below for the cost coefficients do not have any special meaning in and of themselves; it is the relative proportions of the communication and computation costs, μ_{ij} and v_j respectively, that are important. Each of the optimal solutions X^* that appears in the following text satisfies the accuracy requirement of 8% and is used to determine the amount of cost savings generated in comparison to the most cost effective centralized strategy as calculated above in Equation 1.22.

Case One – Communication and processing costs are equal in size and of similar cost:

$$\begin{cases} \mu_{12}=1 & \mu_{13}=1 & \mu_{23}=1 \\ v_1=1 & v_2=1 & v_3=1 \end{cases} \Rightarrow C=\begin{bmatrix} 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix}, \begin{cases} C_0=5 \\ C_1=3 \end{cases} \qquad (1.25)$$

In this situation, we found that the localized solution C_1 provides a cost saving of 40% versus the centralized strategy, which signifies there is a good possibility that a well-balanced combination strategy exists. By using Equation 1.19 as previously mentioned, we reveal an optimal combined strategy:

$$X^* = \begin{bmatrix} 1 & .016 & .081 \\ .073 & 1 & .114 \\ .075 & .038 & 1 \end{bmatrix},\ C^* = 3.79 \qquad (1.26)$$

This combined strategy meets the 8% accuracy requirement we previously set and still provides a 24.1% cost saving compared to the best centralized strategy.

Case Two – Communication and processing costs are of equal size, network communication (data transfer) is more expensive than processing:

$$\begin{cases} \mu_{12}=10 & \mu_{13}=10 & \mu_{23}=10 \\ v_1=1 & v_2=1 & v_3=1 \end{cases} \Rightarrow C = \begin{bmatrix} 1 & 11 & 11 \\ 11 & 1 & 11 \\ 11 & 11 & 1 \end{bmatrix},\ \begin{cases} C_0 = 23 \\ C_1 = 3 \end{cases} \qquad (1.27)$$

Under these circumstances, the localized strategy supplies an 87% cost savings when compared to the best centralized strategy. Our calculation of an optimal strategy using Equation 1.19 shows a 68% cost saving compared to C_0:

$$X^* = \begin{bmatrix} 1 & .016 & .081 \\ .073 & 1 & .114 \\ .075 & .038 & 1 \end{bmatrix},\ C^* = 7.36 \qquad (1.28)$$

This optimal strategy is the same as the strategy in Case One because the relative magnitudes of the non-diagonal coefficients of matrix *C* remained the same so the solution to Equation 1.19 does not change. The component that changed in this situation was the communication cost, which was reduced significantly by the diminished amount of expensive, and avoidable, data transfer.

Case Three – Communication and processing costs are of equal size, data processing is more expensive than network communication (data transfer):

$$\begin{cases} \mu_{12}=1 & \mu_{13}=1 & \mu_{23}=1 \\ v_1=10 & v_2=10 & v_3=10 \end{cases} \Rightarrow C=\begin{bmatrix} 10 & 11 & 11 \\ 11 & 10 & 11 \\ 11 & 11 & 10 \end{bmatrix}, \begin{cases} C_0=32 \\ C_1=30 \end{cases} \qquad (1.29)$$

Again, applying Equation 1.19 results in the same combined strategy:

$$X=\begin{bmatrix} 1 & .016 & .081 \\ .073 & 1 & .114 \\ .075 & .038 & 1 \end{bmatrix}, \; C(X)=34.37 \qquad (1.30)$$

however, this strategy is not optimal; in fact, there is a 7.6% decrease in cost saving when compared to the best centralized strategy and a 14.7% decrease from the best localized strategy.

In these circumstances, it is apparent that the low cost of data transfer permits the relocation of all data to a root node where one extremely accurate model is built. This emerges as a strategy that is both less costly and more efficient than building three local models that are each based on only a fraction of the data. Thus, the optimal strategy is any one of the three equal cost centralized models:

$$X_0(1)=\begin{bmatrix} 10 & 11 & 11 \\ 10 & 11 & 11 \\ 10 & 11 & 11 \end{bmatrix} \quad X_0(2)=\begin{bmatrix} 11 & 10 & 11 \\ 11 & 10 & 11 \\ 11 & 10 & 11 \end{bmatrix} \quad X_0(3)=\begin{bmatrix} 11 & 11 & 10 \\ 11 & 11 & 10 \\ 11 & 11 & 10 \end{bmatrix} \qquad (1.31)$$

Unfortunately, none of these centralized models meet the initial assumption that $x_{ii} = 1$, made above, in the data preparation section. That means Equation 1.19

may not be applied to these strategies and they will need to be analyzed separately; it also explains why using Equation 1.19 did not produce an optimal solution.

Case Four – Communication cost varies and data processing is cheap:

$$\begin{cases} \mu_{12}=10 & \mu_{13}=5 & \mu_{23}=1 \\ v_1=1 & v_2=1 & v_3=1 \end{cases} \Rightarrow C=\begin{bmatrix} 1 & 11 & 6 \\ 11 & 1 & 2 \\ 6 & 2 & 1 \end{bmatrix}, \begin{cases} C_0=9 \\ C_1=3 \end{cases} \qquad (1.32)$$

In this case, we may use Equation 1.19 to calculate the following optimal combination strategy because the model meets the accuracy requirements and there is a perceptible decrease in the amount of network communication between nodes 1 and 2.

$$X^*=\begin{bmatrix} 1 & .004 & .036 \\ .051 & 1 & .190 \\ .105 & .065 & 1 \end{bmatrix},\ C^*=4.96 \quad (\text{a } 45\% \text{ saving over } C_0) \qquad (1.33)$$

Case Five – Both communication cost and data processing cost vary:

$$\begin{cases} \mu_{12}=1 & \mu_{13}=1 & \mu_{23}=10 \\ v_1=10 & v_2=10 & v_3=1 \end{cases} \Rightarrow C=\begin{bmatrix} 10 & 11 & 2 \\ 11 & 10 & 11 \\ 11 & 20 & 1 \end{bmatrix}, \begin{cases} C_0=14 \\ C_1=21 \end{cases} \qquad (1.34)$$

The most cost effective of the three possible centralized strategies is $X_0(3)$ with cost $C_0=14$. The combination strategy we determined by using Equation 1.19

$$X=\begin{bmatrix} 1 & .026 & .246 \\ .073 & 1 & .044 \\ .075 & .031 & 1 \end{bmatrix},\ C(X)=24.5 \qquad (1.35)$$

proves to be more costly than strategy $X_0(3)$ in which all of the data is moved to, and a single model is built at, node three. In this case the localized strategy, usually the most cost effective if accuracy is disregarded, is also more costly than $X_0(3)$. This irregularity develops from a situation in which localized data processing at nodes 1 and 2 is too expensive; transferring all the data to node 3 is a more economical choice. This leads to an optimal strategy $X^* = X_0(3)$.

Many other cases exist that may be analyzed with this framework. Potential modifications that could be explored include:

1. Require the coefficients of communication cost to satisfy triangle inequalities such that $d(v_x, v_z) \leq d(v_x, v_y) + d(v_y, v_z)$ for all v_i in M. This condition may more closely correspond to the option of moving data between sites by way of a third site.
2. Establish additional linear constraints, related to network topology, on the amount of data that is shared between two sites (x_{ij}). This may be necessary in a situation in which the only route between sites 1 and 3 goes through site 2 and it is desirable to make use of all data that is moving from site 1 to site 3 when creating the local model at site 2. In such a case the additional linear constraints would be

$$x_{12} \geq x_{13}\,,\; x_{32} \geq x_{31} \qquad (1.36)$$

   and the optimization procedure would require a few small changes, although the general technique will not require modification.

The examples we outlined above demonstrate that when the structure of the learning process, in this case the error function, is well understood, non-trivial combination strategies are common and frequently are preferable to both localized and centralized strategies.

1.6 Conclusion

We have introduced a new framework for use with distributed data mining in this research. The framework facilitates the development of a cost-optimal balance between localized computation and site-to-site network communication and data transfer. Our methodology successfully combines two common approaches to distributed data mining: one that computes all data in-place, localized mining, and a second, centralized mining, that collects all data at a single site prior to performing any computations. We refer to the mining strategies that are the results of our new framework as combined strategies.

The framework developed in our research reduces the problem of discovering combined strategies to a mathematical optimization problem that minimizes a cost function that includes both network communication and data processing elements and is subject to an error constraint. We presented examples to demonstrate that this optimization problem is interesting even for linear cost functions, and outlined a method for use in discovering combination strategies and determining their relative cost effectiveness.

2 Dual Optimization Strategies

2.1 Introduction

We discussed the challenge related to determining combination strategies for use in distributed data mining in the previous section. Combination strategies introduce a compromise between the accuracy realized by centralized data processing and the cost savings attained by localized processing. We showed that, when the generic form of an error function and the costs related to each phase of data processing are known, the problem of determining a combination strategy may be considered as a problem of minimizing a cost function subject to the desired accuracy levels. In our research, we minimized a linear cost function over a convex feasible set.

A problem that is paired with the situation described above is one of minimizing the error while complying with defined cost conditions:

$$\begin{cases} \min_{[x_{ij}]} E(X) \\ 0 \le x_{ij} \le D_i \\ C(X) = \sum_{ij} c_{ij} x_{ij} \le \Lambda \end{cases} \qquad (2.1)$$

If we adopt the settings used in the previous section, this problem becomes an exercise in minimizing a convex function over a feasible set that is defined by linear inequalities. This is a familiar optimization problem and may be solved using a broad spectrum of methods (Lewis and Borwein, 2000).

There is, however, a fundamental difference between these two problems. The primary issue when minimizing cost is deciding how much data to transfer. The cost function is most affected by differences in the amount of data transmission rather than by changes in data processing costs. In any situation, each optimal combination strategy satisfies the error constraint $E(x) = \varepsilon_{\max}$.

When we minimize the error function E(X) there are typically no local minimums within the interior of the feasible solution, due to the character of E(X), and so the optimal solution to the minimization will be on the boundary, specifically $C(X^*) = \Lambda$. The value of Λ strongly relates to the quantity of data traffic so the debate is no longer over the amount of data to transfer because, knowing the value of Λ, there is little possibility for modification of this figure. As there is no longer a question of the amount of data to transfer, the issue that takes precedence is that of choosing which data to transfer. Selecting different data instances frequently yields different accuracy levels for the predictive models that are built on that data, so that minimizing the error function E(X) when given specific cost constraints correlates more to the problem of selecting the most "suitable" data than to calculating the right amount of data.

2.2 Dual Strategies

In the calculations that follow, we assume that the cost of network communication necessary to support data transfer outweighs the costs related to local data processing and aggregation of the results. This assumption is sensible

because, from a practical standpoint, data delivery and cleaning are the most costly and time-consuming components of the data mining process; data processing is continually proving to be faster and less costly due to the rapid increases in computing power (Pisharath et al, 2006).

Under this assumption, cost is added to a strategy only at the data transfer stage, so the strategy is calculated by the following equation

$$C(X)=\sum_{i\neq j}\mu_{ij}x_{ij}\leq\Lambda, \qquad 1\leq i\leq k \text{ and } 1\leq j\leq n \tag{2.2}$$

where $\mu_{ii} = 0$ and μ_{ij} is the unit cost for data transfer from site *i* to site *j*.

Now, rather than solving the general minimization problem of Equation 2.1, we solve a simplified version of the problem in which the cost is evenly split among all $n(n-1)$ terms in Equation 2.2 such that

$$\begin{aligned} &\min_{[x_{ij}]} E(X) \\ &0\leq x_{ij}\leq D_i \\ &\mu_{ij}x_{ij}\leq\frac{\Lambda}{n(n-1)},\ i\neq j \end{aligned} \tag{2.3}$$

The optimal strategy X* will precisely satisfy the cost condition, as noted above, so we calculate the amount of data to transfer between each pair of nodes as follows:

$$x_{ij}^{*}=\begin{cases}\dfrac{\Lambda}{\mu_{ij}n(n-1)}, & i\neq j\\ 0, & i=j\end{cases} \tag{2.4}$$

The appropriate question at this point is: which data should be selected for this transfer?

2.3 Dual Strategies for Ensemble Learning

In this section of our research we performed experiments to investigate several ways it is possible to build ensembles of local models while influenced by network traffic limitations.

Selecting specific data records for use in a learning algorithm is a technique that several acknowledged supervised learning methods implement, predominantly in the boosting frameworks (Dietterich, 2000). In a conventional boosting strategy, one predictive model is trained many times on data that has been sampled from the same dataset but that has been assigned probability weights that vary with time. A data instance in this boosting scenario has a higher probability of being chosen if the current instance of the training errs in its prediction of the data instance's class value. The size of the error also affects the probability of the choice when the class attribute is a numeric type. This iterative approach guarantees that portions of the initial data distribution in which the model continues to make errors will be more completely disclosed to the learning algorithm in the ensuing iterations while parts of the distribution that are already well learned are not unnecessarily divulged.

Boosting and ensemble methods, or more generally, random forests and alternative random collections of predictive models, are naturally linked (Breiman, 1999), which motivates our research into the possibilities that may arise from using a boosting type technique in the distributed data mining framework outlined above.

The same Forest Cover Type dataset from the UCI KDD Archive as was used in the case study of section 1.5 was used to build the ensembles for this portion of the research. The data was again partitioned into $n = 3$ distributed subsets and we used the GhostMiner® data mining package to create decision trees.

We designed the initial data partition to support investigation into modeling situations in which either all of the data comes from a single source or portions of the data come from several different sources. Because the data used was real and it was impossible to control the source, we initially split the training set either homogeneously or heterogeneously using the methods described below.

To model a *homogeneous* initial split, we randomly divided the data set into n equal parts under the assumption that there is no fundamental difference between the base data distributions at each of the n distributed sites. To create a *heterogeneous* initial partition we built a single C4.5 decision tree on the entire dataset and mimicked the way it split the data. We selected this option because decision trees have the remarkable feature of creating data partitions that are easy to interpret. We sorted the data by its most revealing attribute, the one at the decision tree's root node, and then split it sequentially into n equally sized subsets to be used in our experiments.

We performed each type of experiment three separate times for each of the two initial partition types. We then used five-fold cross-validation to test the accuracy of the collections of local models that were the results of each experiment run. Our test results were averaged across the 15 cross-validation folds. The

accuracy tests were carried out over a range of values for λ, the parameter that defines constraints on network traffic levels. For these experiments, we defined $\lambda \in [0,1]$ to be the highest percentage of data that could be moved between local data subsets.

We outline the experiments we performed for this research below and present the test results in Table 7.

Table 2.1: Misclassification error rates for ensembles built on Forest Cover Type data.

Data Transfer (%)	Same Data Source		Different Data Sources	
	Simple	Boosted	Simple	Boosted
0	5.2	5.2	37.6	37.6
10	5.2	5.5	37.6	37.6
20	5.3	5.6	6.7	5.6
30	5.2	5.5	6.0	5.5
40	5.2	5.5	5.7	5.5
50	5.2	5.4	5.4	5.3
60	5.3	5.6	5.4	5.4
70	5.3	5.3	5.3	5.3

2.3.1 Simple Ensembles

We designed our first type of experiment to provide a base case to which the advantages of alternative methods of data selection may be compared. We revised the initial apportionment of data into *n* subsets, or colors, by performing a random exchange of data for this type of experiment. Specifically, we randomly chose λ percent of each data subset and divided this selection into $n-1$ equally sized portions; each of these portions was then shifted to one of the remaining $n-1$ subsets. We built a C4.5 decision tree model on the freshly collected dataset at

each receiving site. Next, we tested the collection of *n* locally built predictive models as a voting ensemble with a test set defined by the current cross-validation fold.

Because the procedure used to choose the segment of data to be moved during this experiment was fully random, three separate random selections were made for every individual value of λ represented in each of the cross-validation folds. We averaged the test results for these three operations and then further averaged them over the 15 cross-validation folds as described previously.

The Simple Ensemble method, when applied to a random initial data partition in which the colors represent the same data distribution, produces ensembles of essentially the same accuracy, a 5.2% misclassification error, regardless of the data transfer that is allowed. This property is not surprising; it is highly probable that if the initial data partition represents *n* copies of the same data distribution, then random exchanges of data between colors will maintain the local distributions.

There is, however, a significant improvement in the accuracy of the ensemble as the level of data traffic increases in situations where the initial partition represents *n* distinct data distributions. The misclassification error rate of a localized strategy where $\lambda = 0,$ and there is no data movement, is more than seven times higher than the misclassification error rate of a strategy that supports an unlimited mixing of the *n* individual distributions. This is caused by the fact that at the outset, each model of the ensemble is unable to learn any of the remaining $n-1$

distributions and so makes an excessive number of errors on the test set. Polling a vote from such an inferior collection of predictors may result in an error rate of as much as 37.6%. We found that the local models started learning the remaining data distributions as the value λ of the data transfer was increased, and so became increasingly more suitable predictors for the entire dataset, frequently dropping the error rate to as low as 5.3%.

It is noticeable that the satisfactory level of data mixture is realized when the accuracy is equivalent to that of an ensemble built in the instance of a common initial distribution, approximately when the data exchange amount reaches. $\lambda = 60\%$. In fact, the value $\lambda = \frac{k-1}{k} = 66.7\%$ identifies a uniform mixture of the original data distributions, that is, a combination in which each color holds $\frac{1}{n}$ of the data from each of the original distributions.

2.3.2 Boosted Ensembles

In the second set of experiments, rather than randomly choosing data for transfer, we selected boosting as the method of choosing which data occurrences to move. The iterative process that is necessary to use this method in a distributed environment is as follows:

- For iterations $i = 1, 2, \ldots, n$:
  - Build all local models.
  - Define the data transfer constraint, λ_i, for the current iteration.
  - For each pair of data sites (*fromSite*, *toSite*):
    - Choose $\frac{\lambda_i}{n-1}$ percent of data occurrences in the *fromSite* site that are misclassified by the model built on the data of the *toSite* site.
    - Move the selected data instances to the *toSite* node.
- Combine the resulting *n* models into an ensemble that uses a majority vote to make its prediction.

This separates the data into several sets, one for each iteration. The following condition must be fulfilled in order to establish the dependability of this algorithm and the overall data transfer limitation:

$$1 \geq \lambda_1 \geq \lambda_2 \geq \ldots \geq 0, \quad \sum_i \lambda_i \leq \lambda \qquad (2.5)$$

In this implementation of our experiment, $\lambda_i = \frac{\lambda}{2^i}$, which meets the requirement set out in Equation 2.5.

We noticed an improvement in test results when using boosting rather than simple ensembles in situations involving a heterogeneous initial data distribution, that is, different initial distributions. The improvement was most apparent for small or medium levels of allowable data transfer. Exchanging merely 20% of data selected by boosting results in an ensemble was at least as accurate, a 5.6% error rate, or even more accurate than the 5.7% error rate achieved by exchanging twice as much randomly chosen data. As the value of λ increases, the advantages of boosting steadily decrease because when the allowable traffic level is large the

simple ensemble method insures a uniform mixture of the initial local data distributions, leaving little room for improvement.

A similar reason may explain the case in which boosted ensembles were routinely outdone, even if only insignificantly, by simple ensembles when there was a common initial data distribution.

2.4 Dual Strategies for Clustering

Clustering and mixture problems are another class of data mining methods that is based on selecting specific data instances (Everitt, 1974), (Mitchell, 1997). There is a fundamental difference between the voting ensembles of predictive models and those of clustering methods. Every model in an ensemble is expected to learn the whole of the global data distribution. Each cluster in a clustering model, however, corresponds to only a portion of the global distribution that does not usually extend into portions covered by other clusters. In respect to choosing and transferring data instances between clusters, clustering may be thought of as a method opposite to boosting; in boosting, a model is exposed to data on which it makes the largest errors, while in clustering, a cluster is presented with data that fits it better than it fits other clusters.

The focus of this section is on the use of partition based clustering algorithms as the most well suited method to apply to the problem of partitioning data across n distributed sites. For a distributed environment, this is equivalent to considering

each of the *n* distinct sites as a separate cluster and applying the constraints on moving data, as given by Equation 2.4, to the chosen clustering algorithm.

We performed tests for this research using a k-means clustering algorithm, which is a well-known partition based clustering algorithm (Everitt, 1974). To implement this algorithm, we calculated the centroids of our clustering results. The centroid of a cluster is the average point in the multidimensional space defined by the dimensions of that space. In a sense, it is the center of gravity for the respective cluster. It is infeasible to update the cluster centroids too often, as occurs in the standard version of the k-means algorithm, due to the restrictions on the level of data transfer. In fact, if the intention is to maintain the data locally, the strategy would be to exchange only the clustering results, in other words the centroids, between the sites but perform all of the updates locally, which requires maintaining up to date copies of all locally stored centroids. Each cluster may be composed of data from all data sites, thus if the algorithm updates the centroids every time a new data instance, that may come from either of the local sites, is added to the cluster there will be an inordinately frequent exchange of the centroid copies. The data traffic that arises from these exchanges is equivalent to, or perhaps even worse than, the traffic created when the entire dataset is initially centralized.

Updating the cluster centroids by transferring data in batches and only infrequently exchanging the centroids between the *n* sites is one way to overcome this traffic difficulty. There must also be a means of halting the clustering operation before the amount of data transfer exceeds λ. To accomplish this, we

implemented an iterative k-means algorithm similar to the one outlined above for boosting.

- For iterations $i = 1, 2, \ldots, n$:
  - Update the centroids for each color in the n distributed centroids that make up the k-means clustering model.
  - Define the data transfer constraint, λ_i, for the current iteration.
  - For each pair of data sites (*fromSite*, *toSite*):
    - Select $\frac{\lambda_i}{n-1}$ percent of the data instances in the *fromSite* that are closer to the centroid of the *toSite*. Transfer the chosen data instances to the *toSite*.

2.4.1 Clustering Gene Expression Microarray Data

This experiment demonstrates various aspects of the dual strategies for clustering methods. The dataset that we use in this experiment contains gene expression microarray data. Gene expression data is an extremely effective tool for analysis of individual genes and their interactions; it is also a powerful means for discovering natural gene clusters and groups (Gerstein and Jansen, 2000), (Tibshirani et al., 1999). The characteristic microarray dataset is arranged such that rows correspond to individual genes while the columns correspond to specific experiments, environment types, or other external stimuli. The values held in the tables are numerical gene expression values that express the level of activity each gene exhibits in response to each stimulus. Similar genes react to the same stimulus in similar ways so gene expression data is frequently used to cluster genes or determine further information about gene similarity (Gerstein and Jansen, 2000).

The gene expression microarray dataset we selected for use contains the expression values for 2000 individual genes (rows) for 62 different tissues (columns), 40 of which were colon cancer tissues and 22 of which were normal tissues. This dataset was illustrated in (Zhang et al., 2001). Each row represents a separate data instance and each column a data attribute in this investigation into the effect of data transfer constraints on the quality of clustering.

We set the parameters for this experiment such that $n = 10$ nodes and there are $r = 100$ independent runs. Initially we randomly partitioned the data into 10 data subsets. We then ran the distributed k-means clustering algorithm for a range of values for λ, the percentage of allowable data transfer. We selected values of λ that approximately follow a logarithmic scale. In this situation, the value of λ may be greater than 100% because a large overall data transfer results from the property that data may be transferred between local subsets several times before a stable clustering configuration is achieved.

It is generally accepted that low Lp norms such as the Manhattan norm provide better distance metrics than the standard Euclidean norm for high dimensional data (Aggarwal et al., 2001), so the Manhattan distance metric was selected for clustering, and the resulting clusters were inspected. Results of the cluster analysis are presented in Table 8 and the tests themselves are detailed below.

Table 2.2: Analysis of the gene expression microarray data.

Allowable Data Transfer (%)	Actual data transfer (%)	Residual Error (%)	Precision (%)	Recall (%)
0	0	346.09	n/a	n/a
10	11.1	342.31	46.2	14.1
30	23.2	332.81	53.6	28.5
100	61.0	268.87	74.3	52.6
300	113.2	208.39	72.3	76.4
1000	147.8	174.81	88.3	86.8
3000	147.3	164.56	95.5	92.6
10000	157.8	162.22	100.0	100.0

2.4.2 Effects on the Cluster Tightness

The first effect we considered regards how the tightness of the clusters obtained was affected by increasing the amount of allowable data transfer. Each data record may be treated as a point in a 62-dimensional space because all data attributes are numeric. We defined residual error as the average distance between the data point and the nearest of the $n=10$ centroids for each dimension. We then averaged the results over the $r=100$ independent trials to calculate the rates shown in Table 8. We defined the residual error to be the predicted distance as a percentage of the actual distance between a data point and the nearest centroid. This residual error was as high as 346.09% in the initial random distribution, however, consecutive clustering operations increased the tightness of each cluster and, as expected, the average residual error dropped. In our tests, the average residual error dropped to as low as 162.22% when $\lambda=10000\%,$ in other words, when each data record conceivably is moved an average of 100 times. Residual errors started

stabilizing around $\lambda = 3000\%,$ and there was no significant error reduction for values greater than $\lambda = 10000\%.$

It was also noticeable that the actual data traffic did not increase indefinitely as the transfer constraints were loosened, and in fact, traffic leveled out at $150-160\%$ notwithstanding a much greater value for the allowable data transfer level. This stabilization is explained by the circumstance that, for each dataset and task, there is an ideal amount of data transfer necessary to achieve a stable clustering configuration. Data transfer amounts above this ideal level are superfluous and may actually increase residual error levels.

2.4.3 Identification of Similar Genes

A critical question in bioinformatics is finding similar genes, and a goal of this research is to use the results of clustering to solve this problem. In this procedure, a suitable gene, *GeneX*, is chosen from the 2000 genes represented in the data and similar genes are discovered. The identical clustering configuration will not emerge from each run of the algorithm because clustering methods such as k-means depend on the initial conditions; therefore, genes that consistently appear in the same cluster as *GeneX* are identified as the similar genes.

Identification is carried out through the following procedure: Let *GeneY* be another gene from the data. Let $\hat{P}$ be the observed frequency for *GeneY* to occur in the same cluster as *GeneX*, and let *p* be the true frequency of the event. *GeneX*

and *GeneY* will be considered similar if a statistical null hypothesis is rejected: H_0: *GeneX* and *GeneY* appear in the same cluster only randomly.

The null hypothesis and alternative hypothesis are formalized as follows:

$$\begin{aligned} H_0 &: p \le p_0 \\ H_1 &: p > p_0 \end{aligned} \qquad (2.6)$$

where $p_0 = 1/n$ is the frequency of two data records occurring in the same cluster at random when given *n* clusters.

We present a level .05 test for H_0 in the following text. This is a binomial distribution for which a positive outcome is: "Both *GeneX* and *GeneY* are in the same cluster." A binomial random variable may be nearly approximated by a Normal Gaussian distribution when the number of trials $r \ge 100$, and so a Normal Gaussian distribution may be used to approximate the empirical frequency $\hat{P}$ of a binomial distribution (Pugachev, 1984). We now use the Neyman-Pearson Lemma to create a critical region of size .05 for the test. In the case of a Normal random variable the general type of the region is well known: because $\hat{P}$ has a mean value *p* and a variance $p(1-p)/r,$ the .05 size critical region for H_0 is defined by the condition

$$\frac{\hat{P} - p_0}{\sqrt{p_0(1-p_0)/r}} \ge \Phi(.95) = 1.645 \qquad (2.7)$$

where $\Phi(.95)$ represents the 95th percentile of a standard Normal distribution (Hoel et al., 1971).

To determine the genes that consistently occur in the same cluster as *GeneX*, we compute the empirical frequencies $\hat{P}$ for all genes in the pool and then the genes that satisfy Equation 2.7 are selected to make up a stable *GeneX*-cluster.

2.4.4 Effect on Recall and Precision

We completed the experiment outlined above for each value of λ. The quality of the clustering is best when the data transfer restrictions are most lenient, which corresponds to $\lambda = 100$ in this experiment, therefore the *GeneX*-cluster with a value of $\lambda = 100$ is considered to be a true cluster while *GeneX*-clusters with lower λ values are regarded as approximations to the true cluster. This experiment is designed to evaluate the effect of various data transfer limitations on the ability the clustering algorithm has to recover the true *GeneX*-cluster. We investigated the effects on both recall and precision, where recall and precision are defined as:

$$recall = \frac{|Appr \cap True|}{|True|}, \qquad precision = \frac{|Appr \cap True|}{|Appr|} \tag{2.8}$$

where |.| represents the number of elements in the set, and *Appr* and *True* are, respectively, the approximate and true clusters.

Five different genes were randomly selected to fill the role of *GeneX*, which gives five distinct collections of *GeneX*-clusters, in which each collection is built of clusters located under different λ constraints. We averaged the recall and precision values over the five collections and the results are shown in Table 8 above. The first row corresponds to the initial distribution, prior to clustering, so neither recall

nor precision values were gathered. The last row of the table contains both recall and precision values of 100%, representing the true *GeneX*-clusters.

Experimental results indicate that when there is only a small amount of allowable data traffic the value of the recall deteriorates the greatest amount and the clustering method correctly identifies only a small percentage of the genes similar to *GeneX*. The precision is significantly better than the recall though, evidence that the clustering does not select wrong genes as often as it misses the correct genes. This is a result of the fact that the clustering is only capable of producing a small *GeneX*-cluster for low values, and so overlooks many correct gene identifications. In fact, the clustering does not have sufficient time to deviate from a random initial distribution when the data transfer is low, thus the clustering process is so incomplete that at its conclusion many genes occur in essentially random combinations with other genes and the clustering method will completely rearrange those combinations in its next run. Because of this rearrangement, the hypothesis H_0 is rarely rejected and very few gene pairs have the opportunity to be identified as consistently similar.

These results also emphasize the danger related to placing trust in any single run of a clustering method when the primary research interest is in the precise associations between data instances, a risk magnified by the presence of data transfer constraints. The hypothesis tested above is an example of a method that avoids this danger by combining several runs of the clustering procedure.

As the data transfer constraints are relaxed, there is a sharp increase in the value of the recall, a figure that ultimately reaches the precision value. This increase is because pairs of similar genes are found in the same cluster more often as the quality of clustering results improves, causing more frequent rejection of hypothesis H0, the genes' appearance in the same cluster being wholly random. Because of this frequent rejection, the *GeneX*-cluster determined by the clustering method contains more genes, so a smaller number of correct answers are missed, boosting the recall value.

2.5 Conclusion

In this section we established that the quality of the data and data partition is one of the paramount factors to affect the accuracy of data mining systems built in distributed environments. We also demonstrated that it is possible to improve the quality of the data partitions using initial data relocation strategies, assuming that a technique exists to select the correct data instances to transfer. Previously developed learning approaches such as boosting and bagging appear to be useful in developing new methods of marking the data instances, and depending on the data mining task, these approaches may offer a measurable improvement in the quality of the results.

3 Conclusion

Our research work focused on the basic trade off between cost and accuracy in distributed data mining and related strategies for selecting the "right" data instances to transfer. A large collection of research exists in the area of improving data mining accuracy in situations where the entire dataset is directly accessible to the learning algorithm. Independently, many so called localized methods in distributed data mining are based on in-site processing. However, as our work established, an unacceptable loss of accuracy in the localized mining may be caused by the lack of complete information about other portions of the dataset. In contrast, combining all the data at one central location routinely provides the most accurate results, but may be too expensive to implement due to the cost of network traffic, data combination, and algorithmic complexity.

The perspective of our research is that there must be a combination strategy that balances cost and accuracy levels by transferring only a selected subset of the data across the network. The specific characteristics of such a strategy may depend on a number of factors, which prompts the construction of a formal framework that serves as a foundation for developing new methods to identify effective data transfer strategies.

Our research showed that the process of finding these combination strategies may be expressed as a mathematical optimization problem. Two options are available when attempting to balance cost and accuracy: either set a tolerance for

the accuracy and optimize the cost, or set a tolerance for the cost and optimize the accuracy.

Research results we have presented in this thesis demonstrate that in the first case the problem becomes a question of convex optimization and so standard methods may be used to solve it. We also showed this to be useful in the analysis of several interesting situations where combination strategies develop naturally.

In the second case, where the cost is limited and the accuracy is optimized, finding a balanced data transfer strategy involves the areas of data partition quality and mixtures of learning models, large and well-established areas of statistics and data mining. Exploring this link, we developed another mathematical framework, based on the standard Expectation Maximization approach, which formalizes the connection between the trade off in distributed data mining and the mixture of learning models strategy. We used examples to demonstrate how cost constraints may be converted into restrictions in the hidden value space in which interesting likelihood bounds may be discovered, and it also was shown how to apply this framework in specific situations.

Our research also explored other important issues related to the ways data is partitioned for our experimental studies by using data from the UCI KDD Archive and gene expression microarray data.

The question of the trade off between cost and accuracy in distributed data mining is comparatively new so one of the goals of our thesis was to review the issues that arise and to lay a foundation for further research in this area. For this

reason, we explored a formal mathematical treatment of the problem. Our work is designed to permit various interpretations and continued developmental work along these lines by including several *ad hoc* techniques and heuristic methods. We made an attempt to balance rigor and flexibility as these frameworks were developed; this was intended to simplify further research and development efforts.

WORKS CITED

Aggarwal, C.C, Hinneburg, A. and Keim, D.A.: "On the Surprising Behavior of Distance Metrics in High Dimensional Spaces." Proceedings of the International Conference on Database Theory. 2001. 420 – 434.

Breiman, L.: "Random Forests, Random Features." Technical report, University of California, Berkely, 1999.

Chatval, V.: Linear Programming. Freeman and Co., 1983.

Cortes, C., Pregibon, D.: "Signature-based Methods for Data Streams." Data Mining and Knowledge Discovery 5.1 (2001):167 – 182.

Dietterich, T.G.: "Machine Learning Research: Four Current Directions." AI Magazine 18 (1997): 97 – 136.

Dietterich, T.G.: "An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization." Machine Learning 40.2 (2000): 139 – 157.

Everitt, B.: Cluster Analysis. Wiley, New York, NY, 1974.

Gerstein M. and Jansen R.: "The Current Excitement in Bioinformatics, Analysis of Whole-genome Expression Data: How Does it Relate to Protein Structure and Function?" Current Opinion in Structural Biology 10.5 (2000): 574 – 584.

Grimshaw, A.S., Weissman, J.B., West, E.A, and Loyot, E.C.: "Metasystems: An Approach Combining Parallel Processing and Heterogeneous Distributed Computing Systems." Journal of Parallel and Distributed Computing 21.3 (1994): 257 – 270.

Grossman, R. L., Bailey, S., Ramu, A., Malhi, B., and Turinsky, A. "The Preliminary Design of Papyrus: A System for High Performance, Distributed Data Mining over Clusters." Advances in Distributed and Parallel Knowledge Discovery. H. Kargupta and P. Chan, eds. AAAI Press/The MIT Press, Menlo Park, California. 2000. 259 – 275.

Guo, Y., Rueger, S.M., Sutiwaraphun, J., and Forbes-Millott, J.: "Meta-learning for Parallel Data Mining." Proceedings of the 7th Parallel Computing Workshop. 1997. 1 – 2.

WORKS CITED (Continued)

Kantardzic, M., Zurada, J. (eds.): Next Generation of Data-Mining Applications, Wiley-IEEE Press, 2005.

Kargupta, H., Johnson, E., Sanseverino, E.R., Park, B.H., Silvestre, L.D., and Hershberger, D.: "Scalable Data Mining from Vertically Partitioned Feature Space Using Collective Mining and Gene Expression Based Genetic Algorithms." KDD Workshop on Distributed Data Mining, 1999.

Lewis, A.S. and Borwein, J.M.: Convex Analysis and Nonlinear Optimization: Theory and Examples. Cms Advanced Books in Mathematics, Springer Verlag, 2000.

Mitchell, T.: Machine Learning, WCB/McGraw-Hill, 1997.

Murata, T. "Graph Mining Approaches for the Discovery of Web Communities." Proceedings of the First International Workshop on Mining Graphs, Trees and Sequences (MGTS-2003). ECML/PKDD'03 workshop proceedings, September 2003. 79 – 82.

Pisharath, J. et al., "Accelerating Data Mining Workloads: Current Approaches and Future Challenges in System Architecture Design." Proceedings of the 9th International Workshop on High Performance and Distributed Mining. 6th International SIAM Conf. Data Mining, SIAM, 2006. 86 – 88.

Raftery, A.E., Madigan, D., and Hoeting, J.A.: "Bayesian Model Averaging for Linear Regression Models." Journal of the American Statistical Association 92 (1996): 179 – 191.

Ruggieri, S. "Efficient C4.5," IEEE Transactions on Knowledge and Data Engineering. 14.2 (Mar.-Apr. 2002): 438 – 444.

Stolfo, S., Prodromidis, A.L., and Chan, P.K.: "JAM: Java Agents for Meta-learning Over Distributed Databases." Proceedings of the Third International Conference on Knowledge Discovery and Data Mining. AAAI Press. Menlo Park, California. 1997. 74 – 81.

Tibshirani, R., Hastie, T., Eisen, M., Ross, D., Botstein, D. and Brown, P.: "Clustering Methods for the Analysis of DNA Microarray Data." Technical report, Department of Health Research and Policy, Stanford University, 1999.

WORKS CITED (Continued)

Zaki, M. Li, W., and Parthasarathy, S.: "Customizing Dynamic Load Balancing for a Network of Workstations." Journal of Parallel and Distributed Computing: Special Issue on Performance Evaluation, Scheduling, and Fault Tolerance, June 1997. 282.

Zhang, H., Yu, C.Y., Singer, B., and Xiong, M.: "Recursive Partitioning for Tumor Classification with Gene Expression Microarray Data." Proceedings of the National Academy of Sciences of the United States of America. 98.12 (June 5, 2001): 6730-6735.