# Collective Data Mining: A New Perspective Toward Distributed Data Analysis

**Hillol Kargupta, Byung-Hoon Park, Daryl Hershberger, and Erik Johnson**
School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164-2752, USA
{hillol, bhpark, dhershbe, ejohnso1}@eecs.wsu.edu

**Abstract**

This paper introduces the collective data mining (CDM) framework, a new approach toward distributed data mining (DDM) from heterogeneous sites. It points out that naive approaches to distributed data analysis in a heterogeneous environment may result in ambiguous or incorrect global data models. It also notes that any function can be expressed in a distributed fashion using a set of appropriate basis functions and orthogonal basis functions can be effectively used for developing a general DDM framework that guarantees correct local analysis and correct aggregation of local data models with minimal data communication. This paper develops the foundation of CDM, discusses decision tree learning and polynomial regression in CDM for discrete and continuous variables, and describes the BODHI, a CDM-based experimental system for distributed knowledge discovery.

## 1 Introduction

Distributed data mining (DDM) is a fast growing area that deals with the problem of finding data patterns in an environment with distributed data and computation. Although today most of the data analysis systems require centralized storage of data, the increasing merger of computation with communication is demanding data mining environments that can exploit the full benefit of distributed computation. For example, consider the following data analysis applications.

1. Example I: An epidemiologist is interested in finding out the dependency of the emergence of hepatitis-C in U.S. on the weather pattern. She has access to a large hepatitis-C database at the Center for Disease Control and an environmental database at the Environmental Protection Agency. However, they are at two different places and analyzing the data from both of them using a conventional data mining software requires combining the databases at a single location, which may be quite impractical.

2. Example II: Two major financial organizations want to cooperate for preventing fraudulent intrusion into their computing system. They need to share data patterns relevant to fraudulent intrusion. However, they do not want to share the data since it is sensitive. Therefore, combining the databases may not be feasible. Existing data mining systems cannot handle this situation.

3. Example III: A defense organization is monitoring a situation. Several sensor systems are monitoring the situation and collecting data. Fast analysis of incoming data and quick response is imperative. Collecting all the data to a central location and analyzing it there consumes time and this approach may not be scalable for modern situation monitoring systems with large number of different sensors.

4. Example IV: A major multi-national corporation wants to analyze the customer transaction records for quickly developing successful business strategies. It has thousands of establishments through out the world and collecting all the data to a centralized data warehouse, followed by analysis using existing commercial data mining software, takes too long.

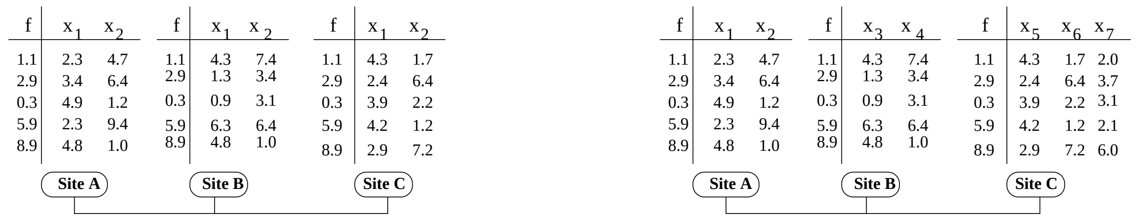
DDM offers a viable solution to many such practical problems. A good DDM algorithm analyzes data in a distributed fashion with modest data communication overhead. Typically DDM algorithms involve local data analysis followed by the generation of a global data model through the aggregation of the local results. Unfortunately, naive approaches to local analysis may produce ambiguous and incorrect global data model. Particularly in the general case, where different sites observe different sets of features, this problem becomes very critical. Therefore, developing a well-grounded methodology to address this general case is important. This paper offers a viable approach to the analysis of distributed, heterogeneous data sets using the *collective data mining* (CDM) framework.

Section 2 describes the DDM problem considered here and some of the problems of naive data analysis algorithms in a DDM environment. In Section 3, the foundation of CDM is presented followed by a discussion on construction of orthonormal representation from incomplete domains and the relation of such representation to mean square error minimization. Sections 4 and 5 present the development of CDM versions of two popular data analysis techniques, decision tree learning and regression. Section 6 presents an overview of a CDM based experimental system called BODHI, that is currently under development. Section 7 concludes this paper and discusses future research directions.

## 2 Background

This section presents the background material for this paper. It first explains the general model of distributed, heterogeneous data sites considered here. Next, a review of related research in DDM is presented. The section concludes by showing that naive approaches to DDM in a heterogeneous environment can be ambiguous or incorrect even for simple data modeling problems.

### 2.1 Problem Description



| f | $x_1$ | $x_2$ |
|---|---|---|
| 1.1 | 2.3 | 4.7 |
| 2.9 | 3.4 | 6.4 |
| 0.3 | 4.9 | 1.2 |
| 5.9 | 2.3 | 9.4 |
| 8.9 | 4.8 | 1.0 |

| f | $x_1$ | $x_2$ |
|---|---|---|
| 1.1 | 4.3 | 7.4 |
| 2.9 | 1.3 | 3.4 |
| 0.3 | 0.9 | 3.1 |
| 5.9 | 6.3 | 6.4 |
| 8.9 | 4.8 | 1.0 |

| f | $x_1$ | $x_2$ |
|---|---|---|
| 1.1 | 4.3 | 1.7 |
| 2.9 | 2.4 | 6.4 |
| 0.3 | 3.9 | 2.2 |
| 5.9 | 4.2 | 1.2 |
| 8.9 | 2.9 | 7.2 |

| f | $x_1$ | $x_2$ |
|---|---|---|
| 1.1 | 2.3 | 4.7 |
| 2.9 | 3.4 | 6.4 |
| 0.3 | 4.9 | 1.2 |
| 5.9 | 2.3 | 9.4 |
| 8.9 | 4.8 | 1.0 |

| f | $x_3$ | $x_4$ |
|---|---|---|
| 1.1 | 4.3 | 7.4 |
| 2.9 | 1.3 | 3.4 |
| 0.3 | 0.9 | 3.1 |
| 5.9 | 6.3 | 6.4 |
| 8.9 | 4.8 | 1.0 |

| f | $x_5$ | $x_6$ | $x_7$ |
|---|---|---|---|
| 1.1 | 4.3 | 1.7 | 2.0 |
| 2.9 | 2.4 | 6.4 | 3.7 |
| 0.3 | 3.9 | 2.2 | 3.1 |
| 5.9 | 4.2 | 1.2 | 2.1 |
| 8.9 | 2.9 | 7.2 | 6.0 |

Figure 1: Distributed (left) *homogeneous* and (right) *heterogeneous* data sites.

In a DDM environment data may be distributed among different sites because of various reasons. For example, the environment may have different databases. On the other hand data may be artificially partitioned among different sites for achieving better scalability. The data sites may be *homogeneous*, i.e. each site stores data for exactly the same set of features. In the general case, however the data sites may be *heterogeneous*, i.e. each site storing data for different sets of features, possibly with *some* common features among the sites. In this document we will consider only the relational model of data. Therefore, the data sets can be viewed as tables with rows and columns. Although in the general case a hierarchy of such relational tables may exist at each site, this paper considers only one table per site in order to build a solid foundation starting from a simpler case. When there are only one table at each site the homogeneous and heterogeneous cases are also sometimes called *horizontally* and *vertically* partitioned. Figure 1 illustrates these two cases. In case of homogeneous sites, each row defines a unique observation. However, in the heterogeneous case different observed features of the same event are distributed among different locations. As a result, the heterogeneous case must come with some predefined way to define the correspondence between the different components of the same event. Row index key and common database schema features are some of the possibilities that may be used to identify the correspondence among different rows stored at different sites.

In this chapter we consider the problem of *supervised learning/analysis for the heterogeneous case.* Given a set of observed feature values, the task is to learn a function that computes the unknown value of a desired feature as a function of other observed features. The given set of observed feature values is sometimes called the training data set. In Figure 1(right) the column for $f$ denotes the feature value to be predicted; $x_1, x_2, x_3, x_4, x_5, x_6$ and $x_7$ denote the features that are used to predict $f$. The data sets available at the different sites are used as the training data. If the $f$ column is not observed everywhere then it is broadcasted to every site. There exists little work for this general case of DDM. The following section reviews related work in the distributed data mining.

## 2.2 Related Work

Although distributed data mining is a fairly new field, it has been enjoying a growing amount of attention since inception. A large fraction of the DDM literature considers homogeneous data sites. Distributed data analysis from homogeneous data sites involves combining different models of data from different sites. There exist several techniques to combine multiple data models. A class of statistical techniques for aggregating multiple models generated from homogeneous data sets is proposed in [2, 3]. The Bagging approach [2] increases the accuracy of the model by generating multiple models from different data sets chosen uniformly with replacement and then averaging the outputs of the models. Although the Bagging technique was initially developed for increasing the accuracy of "unstable" data analysis algorithms, this can be extended to multiple model aggregation in DDM from homogeneous data sets. This perspective is presented in [3].

Like Bagging, Stacking offers an alternate technique to increase the accuracy of the data model by aggregating multiple data models. In Stacking first multiple models are learned on different homogeneous data sets and their joint generalization behavior is observed on a different testing data set. Finally a new model learned using all the data sets and the knowledge about the joint generalization behavior are used for classification or target function prediction. Like Bagging, Stacking can also be extended for combining models in a distributed environment. An experimental investigation of such techniques for combining multiple models is reported elsewhere [40].

Meta-learning [6, 5, 7] offers another class of techniques for mining homogeneous, distributed data. This approach shares a lot of similarities with Bagging and Stacking. In this approach, supervised learning techniques are first used to detect concepts at local data sites, then meta-level concepts are learned from a data set generated using the locally learned concepts, resulting in a meta-classifier. Different inductive learning algorithms may be employed to learn the local concepts, and the meta-level learning may be applied recursively, producing a hierarchy of meta-classifiers. The JAM system [36] is a meta-learning based distributed data mining framework. It has been used for fraud detection in the banking domain [27]. The knowledge probing approach is reported in [14]. This technique is similar to meta-learning. However this approach is particularly designed for inducing descriptive data model from the predictions of black box classifiers learned in a distributed environment.

The Distributed cooperative Bayesian learning approach was developed in [42]. This technique considers homogeneous data sets. In this approach different Bayesian agents estimate the parameters of the target distribution and a population learner combines the outputs of those Bayesian models. The agents and the population learner are based on a probabilistic version of the Gibbs algorithm. The theoretical analysis of the algorithm shows that it performs almost as good as a centralized Bayesian learning algorithm. This paper also proposes a Monte Carlo technique for addressing problems where the hypotheses space is hierarchically parameterized and there is a a feedback mechanism from the population learner to the agent learners.

The *fragmented approach* to mining classifiers from distributed data sources is suggested by [9]. In this method, a single, best, rule is generated in each distributed data source. These rules are then ranked using some criterion and some number of the top ranked rules are selected to form the rule set. In [26] the authors report a technique to automatically produce a Bayesian belief network from discovered knowledge using a distributed approach.

The PADMA system [19, 18] also deals with the problem of distributed data mining from homogeneous data sites. This system implemented a distributed clustering algorithm that was aided by relevance feedback-based supervised learning techniques. The PADMA system has an agent based distributed architecture where the partial data cluster models are first computed in a distributed fashion at different sites; once the individual models are collected to a central site a second level clustering among the different models is performed to generate the overall cluster model. The second level clustering is performed among different clusters by exploiting their statistical representations. A k-means clustering algorithm for distributed environment was reported in [10]. This algorithm notes the inherent data-parallelism in the k-means algorithm and asymptotically approaches near optimal performance.

The Fast Distributed Mining (FDM) algorithm [8] can be used for mining association rules from distributed, homogeneous data sets. FDM notes that in a distributed environment, every globally large itemset must be locally large at one or more sites. It explores this relationship between locally and globally large itemsets in order to minimize the communication overhead. FDM requires $O(s)$ message communication ($s$ is the number of data sites) for determining whether a candidate set is large.

The architecture of a distributed data mining system plays an important role in its performance. Architectural requirements for efficient data communication in a wide area network are explored in [13]. This work also reports several tools like a persistent object manager called PTool, a modeling language called Predictive Model Markup Language (PMML), a model manager called Anubis, and an object transportation layer named Bast to facilitate the local data mining and wide-area combining processes.

A DOALL primitive is proposed in [39] for reducing the complexity of parallel programming in a distributed environment. This work reports scheduling algorithms for assigning tasks to processors to improve load balancing. It also presents empirically evaluation of the DOALL primitive and scheduling algorithms on two-dimensional discretization and clustering problems.

A technique for problem decomposition and local model selection is proposed in [32]. This approach first learns regression models at local data sites. Next it identifies the subset of the data for which the local model works fine. This information is used for partitions the data into different disjoint subsets. Next it learns the distribution functions in order to identify the appropriate local model for each data point. This approach is applied to analyze agricultural data and the authors report substantial improvement in performance compared to a single global model.

There exists very little literature for analyzing data from heterogeneous sites. Learning from heterogeneous data sites is discussed in [33] from the perspective of inductive bias. This work notes that such partitioning of the feature space can be addressed by decomposing the problem into smaller sub-problems when the problem is site-wise decomposable. The WoRLD system [1] addressed the problem of concept learning from heterogeneous sites by developing an "activation spreading" approach. This approach first computes the cardinal distribution of the feature values in the individual data sets. Next, this distribution information is propagated across different sites. Features with strong correlations to the concept space are identified based on this first order statistics of the cardinal distribution. The selected features are used for learning the appropriate concept. Since the technique is based on the first order statistical approximation of the underlying distribution, it may not be appropriate in general for non-convex concept space. The propagation of marker activation records from one site to another is accomplished through basic database operations. This makes the approach easily implementable in database systems. Nevertheless, a general methodology for learning functions from distributed, heterogeneous data sites with guaranteed control of accuracy and minimal communication overhead is still an open issue.

This paper offers one possible solution to this problem. This paper describes the Collective Data Mining methodology that can learn different popular data models such as regression, decision trees in a distributed environment. Interested readers may refer to [15] for additional experimental and theoretical analysis of this framework. The main motivation behind this framework is that direct application of existing machine learning and statistical algorithms to local data sites may produce partials models that are completely incorrect and possibly ambiguous. The following section illustrates this observation.

## 2.3 Naive Approach: May Be Ambiguous And Incorrect

Data modeling is a mature field that has many well-understood techniques in its arsenal. However, many of these traditional techniques cannot be directly used in a distributed environment with vertically partitioned feature space. In this section we shall see that even a simple, decomposable data modeling problem can be ambiguous and misleading in a distributed environment.

Let $f(x_1, x_2) = a_1 x_1 + a_2 x_2$. Consider the data set

$$D = \{(x_1, x_2, f(x_1, x_2)\} = \{(0,0,0), (1,0,a_1), (0,1,a_2), (1,1,a_1+a_2)\}$$

generated by $f(x_1, x_2)$. When both the variables and the corresponding $f(x_1, x_2)$ value are observed at the same site, fitting a linear model of the form $\hat{f}(x_1, x_2) = b_0 + b_1 x_1 + b_2 x_2$ to the data is quite straight forward.

Now consider a distributed environment with two sites, A and B. A observes $\{(x_1, f(x_1, x_2))\}$ and B observes $\{(x_2, f(x_1, x_2))\}$. Consider the dataset at A, $D_A = \{(0,0), (1,a_1), (0,a_2), (1,a_1+a_2)\}$. If site A now

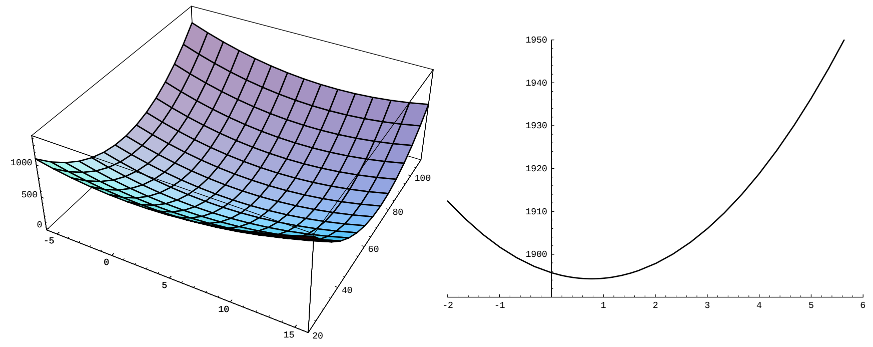


Figure 2: (Left) The global error function. (Right) The local error function at site A.

tries to fit a local linear model of the form $\hat{f}_A(x_1) = b_0^{'} + b_1^{'}x_1$ to the data, then it will get four different solutions of the coefficients, $\{(b_0^{'}, b_1^{'})\} = \{(0, a_1), (0, a_1 + a_2), (a_2, a_1), (a_2, a_1 - a_2)\}$. A similar situation also arises at site B. Resolving these ambiguities requires communication between the two sites. The collective data mining approach offers a solution to this decomposable problem with no communication at all. However, before discussing the CDM let us investigate another possibility: generating local models that minimizes the error between the correct value of $f(x_1, x_2)$ and the models. Unfortunately, as shown in the following, this may also lead to misleading results. Consider the function, $g(x_1, x_2) = 5x_1 + 67.9x_2$, where $x_1$ and $x_2$ are real valued variables. Consider the sample data set $D = \{(1.1, 0.1, 12.29), (1.5, -1.0, -60.4), (-1.9, -0.5, -43.45)\}$, where each entry is of the form $(x_1, x_2, g(x_1, x_2)$. Let us try to fit a model, $\hat{g}(x_1, x_2) = b_1x_1 + b_2x_2$, to this data by minimizing the mean-square error. The overall mean square error computed over the data set $D$ is, $\frac{1}{3}\sum_{x_1,x_2 \in D}(g - \hat{g})^2 = 2.3567(5 - b_1)^2 + 0.42(67.9 - b_2)^2 - 0.1467(5 - b_1)(67.9 - b_2)$

Figure 2 (Left) shows the error surface with a global minima at $b_1 = 5$ and $b_2 = 67.9$. It is a simple quadratic function and the finding the minima is quite straight forward.

Now let us consider the data set to be vertically partitioned; meaning, $x_1$ is observed at site A and $x_2$ is observed at a different site B. Let us choose a linear model, $\hat{g}(x_1) = b_1x_1$. The mean square error function for site A is, $\frac{1}{3}\sum_{x_1,x_2 \in \Omega}(f - \hat{f})^2 = (2.3567(5 - b_1)^2 + 1936.3722 - 19.9173(5 - b_1)$. Figure 2 (Right) shows this local error function. It clearly shows that the minima of this error function is not same as the globally optimal value of $b_1$, i.e. 5. This example demonstrates that even for simple linear data and model, naive approaches to minimize mean-square error may be misleading in a distributed environment. The CDM offers a correct, viable solution to this problem. The following section presents the foundation of CDM.

## 3 The Foundations of Collective Data Mining

Inducing a function structure from data is a common problem in statistics and machine learning. This paper considers distributed function induction from data sets with known function values. In the machine learning literature this is called supervised learning.

In supervised inductive learning, the goal is to learn a function $\hat{f} : X^n \to Y$ from the data set $\Omega = \{(\mathbf{x}_{(1)}, y_{(1)}), (\mathbf{x}_{(2)}, y_{(2)}), \cdots (\mathbf{x}_{(k)}, y_{(k)})\}$ generated by underlying function $f : X^n \to Y$, such that the $\hat{f}$ approximates $f$. Any member of the domain $\mathbf{x} = x_1, x_2, \cdots x_n$ is an n-tuple and $x_j$-s correspond to individual features of the domain. In the heterogeneous case, this would mean learning a function in terms of the features observed at different sites. In most practical situations, distributed inductive learning requires the availability of the $y$-values, i.e. $y_{(1)}, y_{(2)}, \cdots y_{(k)}$ at all the sites. In some applications, the observed data for this particular feature (usually the feature to be predicted or the class label of the events) may be naturally available. If not, then this information may be broadcasted to all the sites. The CDM approaches supervised data analysis and modeling from this perspective of function learning.

## 3.1 Foundations: Blending Theory of Communications with Machine Learning and Statistics

A typical DDM algorithm works by

1. performing local data analysis for generating partial data models, and
2. combining the local data models from different data sites in order to develop the global model.

As we saw in the previous section, conventional approaches for local analysis can be ambiguous and misleading. We need to make sure that the local analysis produces correct partial models that can be used as building blocks for the global model. Once the partial models are generated from the local data sites, the next step is to combine them for generating the global model. However, we need to keep in mind that nonlinear dependency among the features across the different data sites may exist. Therefore, locally generated partial models alone may not be sufficient to generate the global model. The CDM addresses both of these issues as described in the following.

The foundation of CDM is based on the fact that any function can be represented in a distributed fashion using an appropriate basis. Let $\Xi$ be a basis set. Let us index the basis functions in $\Xi$ and denote the $k$-th basis function in $\Xi$ by $\Psi_k$. Let $\Xi_I$ be the set of all such indices of the basis functions. A function $f(\mathbf{x})$ can be represented as,

$$f(\mathbf{x}) = \sum_{k \in \Xi_I} w_k \Psi_k(\mathbf{x}) \tag{1}$$

where $\Psi_k(\mathbf{x})$ denotes the k-th basis function and $w_k$ denotes the corresponding coefficient. The objective of a learning algorithm can be viewed as the task to generate a function, $\hat{f}(\mathbf{x}) = \sum_{k \in \hat{\Xi}_I} \hat{w}_k \Psi_k(\mathbf{x})$, that approximates $f(\mathbf{x})$ from a given data set; $\hat{\Xi}_I$ denotes a subset of $\Xi_I$; $\hat{w}_k$ denotes the approximate estimation of the coefficients $w_k$. For a given basis representation, the underlying learning task is essentially to compute the non-zero, significant (not negligible) coefficients, $\hat{w}_k$-s. Any inductive data modeling algorithm like regression, decision trees and neural networks can be viewed from this perspective. The CDM notes that although direct application of these modeling algorithms may not produce correct results, the orthonormal spectrum of such models can be accurately learned in a distributed fashion and then finally the model can be constructed from the spectrum. The orthonormality property guarantees correct and independent local analysis that can be used as a building-block for the global model. The main steps of the CDM approach are,

1. Choose an appropriate orthonormal representation for the type of data model to be built. For example, our preliminary investigation in this paper and [15] noted the suitability of Fourier and Wavelet representations for decision tree and multi-variate regression respectively.
2. Construct the orthonormal representation from the data in a distributed fashion.
3. Construct the model in its canonical representation (e.g. a tree in case of a decision tree) from the orthonormal representation

Both continuous and discrete functions can be learned in this fashion. However, in the following discussion we shall use only discrete functions for explaining the basic concepts. The following sections explain the basic mechanism of CDM using a simple example.

### 3.1.1 Generating Correct Partial Models Using Only Local Features

Consider a quadratic data modeling problem in which the data set is generated by the function $f(x_1, x_2) = a_1 x_1 + a_2 x_2 + a_3 x_1 x_2$, where $x_1$ and $x_2$ are boolean variables. Note that the function is not site-wise decomposable. The data set is $D = \{(x_1, x_2, f(x_1, x_2)\} = \{(0,0,0), (1,0,a_1), (0,1,a_2), (1,1,a_1 + a_2 + a_3)\}$. As before, sites A and B observe $x_1$ and $x_2$ respectively. Although the naive approach faces an ambiguous situation, no such problem exists if we use orthonormal basis functions for modeling the data. For example, let us consider discrete Fourier basis representation. In this case, $\Xi_k = \{00, 10, 01, 11\}$ (i.e., the set of all 2-bit strings). Any function $f(\mathbf{x})$ of boolean variables can be written as $f(\mathbf{x}) = \sum_{\mathbf{j} \in \Xi_k} w_{\mathbf{j}} \Psi_j(\mathbf{x})$, where $\Psi_{\mathbf{j}}(\mathbf{x})$ denotes the $\mathbf{j}$-th Fourier basis function and $w_{\mathbf{j}}$ denotes the corresponding Fourier coefficient; for this case $\mathbf{x} \in \{00, 10, 01, 11\}$

and $\Psi_{\mathbf{j}}(\mathbf{x}) = (-1)^{\mathbf{j}.\mathbf{x}}$. Fourier coefficients are defined as $w_{\mathbf{j}} = \frac{1}{N}\sum_{\mathbf{x}} f(x)\Psi_j(\mathbf{x})$, where $N = 4$ is the total number of members of the domain. There are four Fourier basis functions in a function of two variables: $\Psi_{00}(\mathbf{x}) = 1, \Psi_{10}(\mathbf{x}) = (-1)^{x_1}, \Psi_{01}(\mathbf{x}) = (-1)^{x_2}$ and $\Psi_{11} = (-1)^{x_1+x_2}$. Clearly, computation of $w_{00}$ does not require either of $x_1$ and $x_2$; computation of $w_{10}$, $w_{01}$, $w_{11}$ requires *only* information concerning $x_1$, $x_2$, or $x_1, x_2$ respectively. Using the above definitions, we get $w_{10} = -(2a_1 + a_3)/4$ from site A and $w_{01} = -(2a_2 + a_3)/4$ from site B; $w_{00} = (2a_1 + 2a_2 + a_3)/4$ can be computed at either site A or site B since computation of $w_{00}$ does not require the feature values.

We can easily demonstrate that the locally generated Fourier coefficients represent partial models that can be finally put together for generating the correct global data model. In order to show that, let us use the locally generated model at site A, $\hat{f}_a(x_1) = w_{00} + w_{10}(-1)^{x_1}$, to generate a data set $D_a = \{x_1, \hat{f}_a(x_1)\} = \{(0, w_{00} + w_{10}), (1, w_{00} - w_{10})\}$. Similarly at site B we generate the dataset $D_b = \{x_2, f_b(x_2)\} = \{(0, w_{01}), (1, -w_{01})\}$ using the model $f_b(x_2) = w_{01}(-1)^{x_2}$. Solving a set of linear equations we generate the partial models in the canonical representation, $y = b_0 + b_1 x_1$ at site A. This results in the local model $y_a$ an't site A, and a similarly generated model $y_b$ at Site B, where:

$$\begin{aligned} y_a &= \frac{a_2}{2} + \frac{2a_1 + a_3}{2} x_1 \\ y_b &= -\frac{2a_2 + a_3}{4} + \frac{2a_2 + a_3}{2} x_2 \end{aligned}$$

Before combining these partial models we need to generate the non-linear term involving the features from both site A and B. The following section describes that process.

### 3.1.2 Generating Cross-Terms Involving Features From Different Sites

If the model is not completely decomposable among the data sites, *cross-terms* involving features from different sites will exist in the model. Determination of these cross-terms is essential for accurate modeling of the data. For the particular example at hand, there exists one coefficient $w_{11}$ that corresponds to $\Psi_{11}(\mathbf{x})$, a basis function requiring both $x_1$ and $x_2$. The coefficient $w_{11}$ can be computed using the feature values of a single a row from all the datasets. In general if there are $m$ cross-terms involving features from different sites, for a well-posed problem solving the $m$ terms requires exactly $m$ rows from all the data sites. For large databases, $m$ is typically a much smaller number compared to the total number of rows in the databases. For example, let us bring the $x_2 = 0$ information from the first data row to site A in order to have a complete row of data $\{x_1 = 0, x_2 = 0, f(x_1, x_2) = 0\}$. After combining the locally generated partial models we get the following expression, $\hat{f}(\mathbf{x}) = 0 = w_{00} + w_{10} + w_{01} + w_{11}$. Since, we already know all the coefficients except $w_{11}$, it can be easily solved to get $w_{11} = a_3/4$. As we see, the Fourier representation of a nonlinear model can be generated by moving only a small fraction of data. If the cost of moving a data row from one site to the common site computing the cross-terms is $c$, then the cost of the communication is $O(c\ m)$. Since many of the real-life interesting problems exhibit bounded non-linearity, i.e. at most some $\kappa$ variables non-linearly interact with each other, $m$ is expected to be a small number compared to the typical number of rows in a large database.

Now the partial model involving the cross-term $w_{11}$ can be used to generate the remaining part of the global model in the canonical representation. Since this term involves both $x_1$ and $x_2$, the general model for this part takes the form $y = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_1 x_2$. We can generate the the dataset $D_c = \{x_1, x_2, f_c(x_1, x_2)\} = \{(0, 0, \frac{a_3}{4}), (1, 0, -\frac{a_3}{4}), (0, 1, -\frac{a_3}{4}), (1, 1, \frac{a_3}{4})\}$ using the function $y_c = w_{11}(-1)^{x_1 x_2}$. The resulting model in the canonical representation turns out to be:

$$y_c = \tfrac{a_3}{4} - \tfrac{a_3}{2} x_1 - \tfrac{a_3}{2} x_2 + a_3 x_1 x_2$$

Now we can combine all the locally generated models in the canonical representation, resulting in:

$$y = y_a + y_b + y_c = a_1 x_1 + a_2 x_2 + a_3 x_1 x_2.$$

This illustrates that locally generated Fourier coefficients and the cross-terms computed using only $O(c\ m)$ communication cost can be effectively used for distributed data modeling. However, there is one final issue that we need to address. So far, our discussion considered a data set that is exactly the complete domain. Typically, learning and data analysis is performed on a sample of the domain, not the complete domain. This would require efficient estimation of the coefficients from incomplete knowledge about the domain. The following section addresses this issue.

## 3.2 Orthonormal Representation Construction from Incomplete Domain

Construction of an orthonormal representation requires computation of the basis coefficients. The exact procedure for computing these coefficients depends upon the specific set of chosen basis functions. Different specialized "fast" techniques exist for computing different orthonormal representations. However, regardless of the specific choice, a function with a large number of significant basis coefficients will require significant time for computing the orthonormal representation. Polynomial time computation of the coefficients requires two things: (1) a sparse representation, where most of the coefficients are zero or negligible, and (2) approximate evaluation of the significant coefficients. Fortunately, this requirements has deep connection to the foundation of search, machine learning, and optimization. For most practical data mining applications these conditions can be satisfied without sacrificing the quality of the process.

Typically, one of the main objectives of a data mining application is to capture a salient data pattern in a simple, easy-to-understand representation. For example, in a classification problem we want not too complex rules that are fairly accurate and covers a reasonable portion of the event space. If we are trying to build a predictive model using polynomial regression techniques, then for all practical purposes we would like to have a model with the degree of polynomial bounded by some constant. Even for black-box learning techniques such as neural networks, specialized, restricted representations are often required for efficient polynomial-time learning. A simple decision tree learning algorithm like the ID3 [34] may engage in an exponential-time tree-construction in the worst case unless we impose a restriction on the depth of the tree. For most practical applications of a decision tree, we want a tree of short depth since they give rise to simple rules. If the performance of the short tree is not good then we may conclude that the decision tree may not be a good representation for the given learning problem. Such restriction of representation is certainly a trade-off. It imposes an algorithmic bias and therefore it restricts the scope of the algorithm. However, it is also often necessary from the perspective of efficiency and efficacy.

Fortunately such restriction on the data model is also reflected in its orthonormal representation. For example, a bounded depth decision tree has a very sparse Fourier representation with only a polynomial number of non-zero coefficients [25]. Moreover, as we will see later in this chapter, some of these coefficients are exponentially larger than the rest of the coefficients. This essentially means that the Fourier representation of a decision tree can be approximated by a very small number of non-zero coefficients. Similar properties show up for decision rules involving only a bounded number of features. If the dependency among the different features is bounded, i.e. if only a constant number of features may depend on a particular feature then the orthonormal representation is going to be sparse.

Estimation of orthonormal coefficients is something that is done on a regular basis in many daily appliances that require signal-processing. If our sample size is reasonable (which is typically the case in data mining) this can be adequately addressed.

Let us illustrate the rationale behind this observation using our Fourier basis example. Consider what happens when we multiply both sides of Equation 1 by $\Psi_{\mathbf{j}}(\mathbf{x})$; we get $f(\mathbf{x})\Psi_{\mathbf{j}}(\mathbf{x}) = \sum_{k\in\Xi_I} w_k\Psi_k(\mathbf{x})\Psi_{\mathbf{j}}(\mathbf{x})$. If we denote our sample data set by $\Gamma$, then by summing both side over all members of $\Gamma$ we get:

$$\sum_{\mathbf{x}\in\Gamma} f(\mathbf{x})\Psi_{\mathbf{j}}(\mathbf{x}) = \sum_{\mathbf{x}\in\Gamma}\sum_{\mathbf{k}\in\Xi_I} w_{\mathbf{k}}\Psi_{\mathbf{k}}(\mathbf{x})\Psi_{\mathbf{j}}(\mathbf{x}) \tag{2}$$

Now note that since $\Psi_{\mathbf{j}}(\mathbf{x})\Psi_{\mathbf{j}}(\mathbf{x}) = 1$, we get: $\sum_{\mathbf{x}\in\Gamma}\Psi_{\mathbf{j}}(\mathbf{x})\Psi_{\mathbf{j}}(\mathbf{x}) = |\Gamma|$, where $|\Gamma|$ is the sample size. Now we can write,

$$\frac{1}{|\Gamma|}\sum_{\mathbf{x}\in\Gamma} f(\mathbf{x})\Psi_{\mathbf{j}}(\mathbf{x}) = w_{\mathbf{j}} + \sum_{\mathbf{k}\in\Xi_I,\mathbf{k}\neq\mathbf{j}} w_{\mathbf{k}}\frac{\sum_{\mathbf{x}\in\Gamma}\Psi_{\mathbf{k}}(\mathbf{x})\Psi_{\mathbf{j}}(\mathbf{x})}{|\Gamma|}$$

Since $\sum_{\mathbf{x}\in\Gamma}\Psi_{\mathbf{k}}(\mathbf{x})\Psi_{\mathbf{j}}(\mathbf{x})/|\Gamma|$ is the sample mean, and the population mean over the complete domain is zero, the sample mean must approach zero as the sample size increases. Therefore, for a large sample size, which is typically the case for data mining problems, the last term should approach zero. As a result, Fourier coefficients computed over large enough samples should approximate the exact coefficients well.

Approximate computation of basis coefficients can be made further efficient using different techniques. One possibility is to group the coefficients into different equivalence classes over the space of all indices and then estimating this individual groups. Consider the classes, $w_{0\#} = \{w_{00}, w_{01}\}$, $w_{1\#} = \{w_{10}, w_{11}\}$; the # character represents a wild card. Define, $S_\alpha = \sum_\beta w_{\alpha\beta}^2$; where $\alpha$ and $\beta$ denote similarity based equivalence classes defined over the first $|\alpha|$ and the last $|\beta|$ values of the index respectively; 0# and #1 are examples

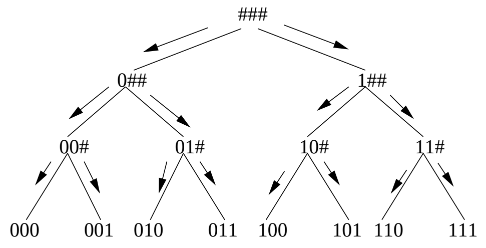


Figure 3: Flow of the $S_\alpha$ computation for different $\alpha$ values.

of $\alpha$ and $\beta$ respectively. $\alpha\beta$ denotes the intersection of classes $\alpha$ and $\beta$. For example, $S_{0\#} = w_{00}^2 + w_{01}^2$. Now note that if any of the individual $w_{\alpha\beta}$-s has a magnitude greater than some threshold value $\theta$, then $S_\alpha$ must have a value greater than $\theta^2$. Therefore, if $S_\alpha < \theta^2$, then none of the Fourier coefficients with an index string starting with $\alpha$ has a significant magnitude. Figure 3 schematically illustrates the flow of the algorithm. At every node of the tree we approximately compute $S_\alpha$, and if $S_\alpha$ at the $i$-th node is less than $\theta^2$ then none of its children can have an $S_\alpha$ value greater then $\theta^2$ and therefore the subtree can be discarded. If the number of non-zero Fourier coefficients is bounded by a polynomial (recall our assumption of bounded non-linearity), we should be able to discard many such sub-trees just by checking the $S_\alpha$ at the root of the sub-tree. Using this idea, a polynomial time algorithm has been developed elsewhere [25] for learning boolean functions with sparse Fourier representation. Also, note that the intersection of all the locally detected significant $\alpha$-s defines a superset of all the indices corresponding to significant cross-terms. This can be used for efficiently approximating the orthonormal representation. In the recent past [23, 22]proposed several randomized and deterministic algorithm for efficiently constructing the Fourier representation.

When the underlying data sampling process is biased and the available data set is not large, alternate techniques may be used to estimate the coefficients. One may use the appropriate orthonormal representation and use standard error minimization algorithm such as least-square minimization technique for fitting the orthonormal coefficients to the available data. This approach will be illustrated in a later section for detecting decision rules.

Earlier, in Section 2, we saw that minimizing the mean-square error of the local model can lead toward incorrect results. The following section shows that the orthonormal basis function based representation does not suffer from this problem.

### 3.3 Orthonormal Representation and Mean Square Error

The search for an appropriate data model can be posed as a model-error minimization problem. In this section we show that minimization of the error of a local model leads toward a correct partial model that can be used as a building-block of the correct global data model. Consider:

$$\begin{aligned} f - \hat{f} &= \sum_k (w_k - \hat{w}_k)\Psi_k(\mathbf{x}) \\ (f - \hat{f})^2 &= \sum_{j,k} (w_j - \hat{w}_j)(w_k - \hat{w}_k)\Psi_j(\mathbf{x})\Psi_k(\mathbf{x}) \end{aligned}$$

Now summing it over all the data points in the training set $\Gamma$:

$$\sum_{\mathbf{x}\in\Gamma} (f - \hat{f})^2 = \sum_{j,k} (w_j - \hat{w}_j)(w_k - \hat{w}_k) \sum_{\mathbf{x}\in\Gamma} \Psi_j \Psi_k$$

Where, $\Psi_j$ is the abbreviated representation of $\Psi_j(\mathbf{x_i})$. Note that the basis functions are assumed to be orthonormal. Therefore, $\sum_{\mathbf{x}} \Psi_j(\mathbf{x})\Psi_k(\mathbf{x}) = 0$, when the sum is over all $x$-s in the space under consideration and $j \neq k$. On the other hand $\sum_{\mathbf{x}} \Psi_j(\mathbf{x})\Psi_j(\mathbf{x}) = 1$. Let us define a random variable $Z_i = \Psi_j(\mathbf{x_i})\Psi_k(\mathbf{x_i})$. Now $E[Z_i] = \sum_{\mathbf{x}_i} \Psi_j(\mathbf{x_i})\Psi_k(\mathbf{x_i}) = 0$, when $j \neq k$. By the law of large numbers $\frac{\sum_{\mathbf{x}_i \in S} Z_i}{n}$ approaches $E[Z_i] = 0$ as

$n$ increases. Therefore for large $n$, we can write,

$$\sum_{\mathbf{x}\in\Gamma}(f-\hat{f})^2 \quad = \quad \sum_{j}(w_j-\hat{w}_j)^2 \tag{3}$$

Clearly, the overall sum of square error is minimized when $\hat{w}_j = w_j$ for all $j$. This derivation assumes that all the feature variables are observed and available for model building at the same time. Let us now investigate if the situation changes when feature space is vertically partitioned. Let us assume that the feature space is divided into two sets A and B with feature spaces $S_a$ and $S_b$ respectively. $\Xi_a$ and $\Xi_b$ be the set of all basis functions defined by feature variables in $S_a$ and $S_b$ respectively; $\Xi_{ab}$ be the set of those basis functions in $\Xi$ that use feature variables from both $S_a$ and $S_b$. Therefore $\Xi = \Xi_a \cup \Xi_b \cup \Xi_{ab}$. We write $j \in \Xi_a$ to denote a basis function $\Psi_j(\mathbf{x}) \in \Xi_a$; we also write $j \notin \Xi_a$ to denote a basis function $\Psi_j(\mathbf{x}) \in \Xi_b \cup \Xi_{ab}$. Now let us explore what happens when one of these sites try to learn $f(\mathbf{x})$ using only its local features. Let us define:

$$\hat{f}_a(\mathbf{x}) = \sum_{j\in\Xi_a} \hat{w}_j \Psi_j(\mathbf{x}) \tag{4}$$

From Equations 1 and 4 we can write:

$$f(\mathbf{x}) - \hat{f}_a(\mathbf{x}) = \sum_{j\in\Xi_a}(w_j-\hat{w}_j)\Psi_j(\mathbf{x}) + \sum_{j\notin\Xi_a} w_j\Psi_j(\mathbf{x})$$

Using the above equation we can write:

$$\begin{aligned}(f(\mathbf{x})-\hat{f}_a(\mathbf{x}))^2 \quad = \quad & \sum_{i,j\in\Xi_a}(w_i-\hat{w}_i)(w_j-\hat{w}_j)\Psi_i\Psi_j + \\ & \sum_{i\notin\Xi_a, j\in\Xi_a} w_i(w_j-\hat{w}_j)\Psi_i\Psi_j + \\ & \sum_{i\in\Xi_a, j\notin\Xi_a} w_j(w_i-\hat{w}_i)\Psi_i\Psi_j + \\ & \sum_{i\notin\Xi_a, j\notin\Xi_a} w_i w_j \Psi_i\Psi_j\end{aligned}$$

Now again using the law of large number we can write:

$$\sum_{\mathbf{x}\in\Gamma}(f-\hat{f}_a)^2 \quad = \quad \sum_{i\in\Xi_a}(w_i-\hat{w}_i)^2 + \sum_{j\notin\Xi_a} w_j^2 \tag{5}$$

Equation 5 tells us that $\sum_{\mathbf{x}\in\Gamma}(f-\hat{f}_a)^2$ takes the minimum value of $\sum_{j\notin\Xi_a} w_j^2$ when $\hat{w}_j = w_j$. Although the minimum value of the error is non-zero, this optimal solution value of $w_i$, $\forall i \in \Xi_a$ remains correct in the global context, even when all the features are considered together. The only difference between the global learning and local learning process is the error term $\sum_{j\notin\Xi_a} w_j^2$ introduced by the basis functions defined by the feature variables not observed at site A.

Although our discussion so far considered only boolean features, the CDM is certainly not restricted to such cases. The following section briefly discusses this issue.

## 3.4 Non-Binary Features and CDM

Orthonormal representations for both non-binary discrete and continuous valued features can also be computed for extending CDM to these domains. There exists many choices of orthonormal basis functions that can handle these cases. For example, discrete Fourier functions can be easily extended to $\lambda$-ary features (a feature can take $\lambda$ different values).

$$\psi_{\mathbf{j}}^{(\lambda)}(\mathbf{x}) \quad = \quad \exp^{\frac{2\pi i(\mathbf{x}.\mathbf{j})}{\lambda}} \tag{6}$$

Where $\mathbf{j}$ and $\mathbf{x}$ are $\lambda$-ary strings of length $\ell$. In other words $\mathbf{j} = j_1, j_2, \cdots j_\ell$ and $\mathbf{x} = x_1, x_2, \cdots x_\ell$. The set of all $\psi_j(\mathbf{x})$ for all possible $\lambda$-ary strings $\mathbf{j}$ defines a basis.

Wavelet representation [41] is a possible choice of orthonormal basis functions for dealing with continuous valued features. We shall discuss these possibilities in detail in later sections. The following section identifies the overall CDM algorithm.

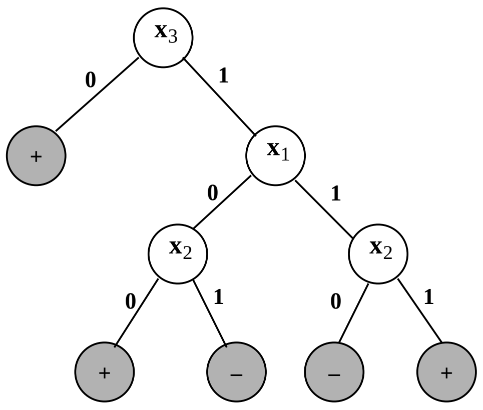


Figure 4: Boolean Decision Tree

### 3.5 The CDM Framework

The CDM is a framework that offers a methodology to develop distributed data analysis algorithms from heterogeneous sites. The main steps of this approach may be summarized as follows:

1. choose an orthonormal representation that is appropriate for the type of data model to be constructed;
2. generate approximate orthonormal basis coefficients at each local site;
3. if necessary, move an appropriately chosen sample of the datasets from each site to a single site and generate the approximate basis coefficients corresponding to non-linear cross terms;
4. combine the local models, transform the model into the user described canonical representation, and output the model.

Development of specific instances of CDM for different data mining techniques is currently on-going. The coming sections will present a flavor of that for two popular data mining techniques. The following section presents the CDM versions of rule-learning using decision tree .

## 4 Decision Tree Learning In CDM

Decision trees [34] are popular techniques for learning classifiers. In this section we describe the ongoing research on a CDM-based approach to construct both numeric and symbolic decision trees from heterogeneous data sites.

### 4.1 An Overview Of the ID3 Algorithm

The ID3 algorithm builds a decision tree from a given labeled data set. For the sake of simplicity, let us consider a boolean decision tree as depicted in Figure 4. The boolean class labels correspond to positive and negative instances of the concept class. We can express boolean decision tree as a function $f\ : X^{\ell} \to\ \{0,1\}$. The function $f$ maps positive and negative instances to one and zero respectively. A node in a tree is labeled with a feature $x_i$. A downward link from the node $x_i$ is labeled with an attribute value of $i$-th feature. A path from the root node to a successor node represents the subset of data that satisfies the different feature values labeled along the path. These data subsets are essentially similarity based equivalence classes and we shall call them schemata (schema in singular form). If $\mathbf{h}$ is a schema, then $\mathbf{h} \in \{0,1,*\}^{\ell}$, where $*$ denotes a wild-card that matches any value of the corresponding feature. For example, the path $\{(x_3 \overset{1}{\to} x_1, x_1 \overset{0}{\to} x_2\}$ in

Figure 4 represents the schema $0*1$, since all members of the data subset at the final node of this path take feature values 0 and 1 for $x_1$ and $x_3$ respectively.

The ID3 algorithm builds the tree by first constructing the root node using the following scheme. It computes the *information-gain* for each variable and assigns the variable to the root that maximizes the expected information-gain. Given the schema $\mathbf{h}$ for the path to a node from the root, the information gained by replacing the wild card value of some feature $x_i$ in $\mathbf{h}$ can be defined as,

$$Gain(\mathbf{h}, x_i) = Entropy(\mathbf{h}) - \sum_{v \in Values(x_i)} \frac{|\mathbf{h}_v|}{|\mathbf{h}|} Entropy(\mathbf{h}_v)$$

where $Values(x_i)$ is the set of all possible values for attribute $x_i$, and $|\mathbf{h}_v|$ is the set of those members of $\mathbf{h}$ that have a value $v$ for attribute $x_i$; if $p$ and $q$ are the proportions of the positive and negative instances in some $\mathbf{h}$, then $Entropy(\mathbf{h}) = -p\log p - q\log q$. Once a variable is assigned to the root, the ID3 algorithm computes information gain for its children and continues to add new nodes if the information gain is significant. As we see, computation of entropy for a given schema is what we need for a decision tree construction. Computation of the entropy at each node requires the knowledge of the distribution of class labels for the data set. In a heterogeneous environment, comparing the information gain requires exchange of distribution information among the different sites. The following section points out that the naive approach to do this by synchronous message exchange may not be scalable.

## 4.2 Naive Approach May Not Scale-up

Decision trees constructed in an independent and asynchronous fashion at different *heterogeneous data sites* may not be combined to generate the correct tree that a centralized construction would produce. An alternate approach to guarantee correctness is synchronous construction where all the sites contribute in selecting a feature at every node of the globally maintained tree. However, the latter approach may not be scalable when the number of sites grows. Since at every node of the tree ID3 computes the information gain for every choice of feature, all the sites need to be informed regarding the particular subset of data subsumed at a particular node. Since the complete data set is distributed among all the nodes at a particular level, at each level the communication cost will be $O(n*s)$, where $n$ is the number of data rows and $s$ is the number of different data sites. If the tree has a depth bounded by some constant $k$ then the overall communication cost will be $O(n*s*k)$. If the table has $c$ columns and $n$ rows then the cost of moving the complete data sets to a single site is $O(n*c)$. When $s*k > c$ distributed decision tree construction using the naive approach is computationally worse than the centralized approach. For distributed environments with large number of data sites this may be a major bottle-neck.

## 4.3 Fourier Spectrum of a Decision Tree

The CDM-approach for distributed decision tree learning is based on a new perspective of decision tree construction. It notes that a decision tree defines a function over the domain under consideration. Even if the features are symbolic, the tree still defines a function. If we define a table that replaces the symbolic values of a feature by some numeric value then we can transform the symbolic function to a numeric function. Therefore we can always compute the Fourier Spectrum of a decision tree. In this section, we shall first consider boolean decision trees and then we shall gradually introduce the CDM approach toward non-boolean decision tree. For almost all practical purposes, decision trees should have a bounded depth. It turns out that the Fourier representation of a decision tree with bounded depth has some very interesting properties [25] that can be quite useful for its distributed construction. These observations are discussed in the following:

1. Decision trees with bounded depth are normally useful for data mining purposes.

2. The Fourier representation of a bounded depth (say $k$) boolean decision tree only has a polynomial number of non-zero coefficients; all coefficients corresponding to partitions involving more than $k$ feature variables are zero.

3. If the order of a partition be its number of defining features then the magnitude of the Fourier coefficients decay exponentially with the order of the corresponding partition; in other words low order coefficients are exponentially more significant than the higher order coefficients.

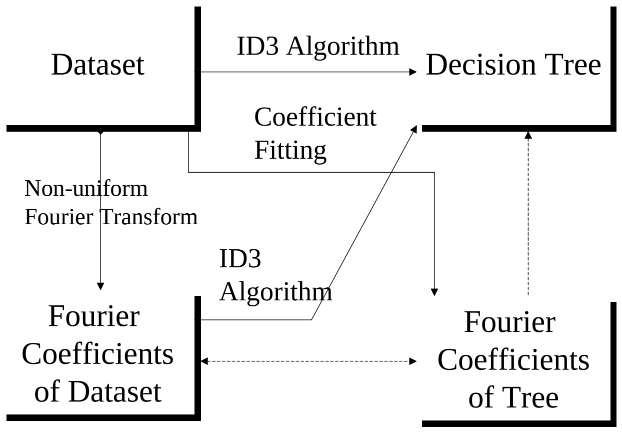

Dataset
ID3 Algorithm
Decision Tree
Coefficient Fitting
Non-uniform Fourier Transform
ID3 Algorithm
Fourier Coefficients of Dataset
Fourier Coefficients of Tree

Figure 5: From data to decision tree through Fourier analysis.

The above two theorems rigorously proves that the spectrum of the decision tree can be approximated by computing only a small number of low-order[1]. The proposed CDM-based approach exploits this observation. In stead of moving all the data to a single data, this approach simply computes the low-order Fourier coefficients from the distributed sites, collects them to a single site, and use them to generate the decision rules. The main strength of this approach is fundamentally based on the suitability of the Fourier basis in representing a decision tree of bounded depth.

Although, the Fourier spectrum of a decision tree is sparse and it can be approximated by only the low-order coefficients, computation of these coefficients from the data set is slightly tricky. This is simply because the training data set is often only a small subset of the complete domain. Naive computation of the Fourier coefficients using only the sample data set may not give us the Fourier spectrum of the tree. Accurate computation of the spectrum from the sample data set will require appropriate estimation. However, before describing such a technique let us discuss the big picture that captures the course from data to the tree through Fourier analysis. The following section does that.

## 4.4 From Data to Decision Tree: One Goal but Different Roads

The CDM approach constructs the decision tree from the data set through Fourier analysis. Figure 5 presents a schematic diagram of the different possible ways to do this. First of all note that the computation of the Fourier coefficients require all the members of the domain (refer to section 3.1.1). However, the learning data set typically does not have that. It turns out that if we assume the class label ($f(\mathbf{x})$) to be 0 for all the members of the domain that are not in the learning data set then the Fourier spectrum exactly represents the data. This is called the Non-uniform Fourier Transformation (NFT) of the data [4]. This will be explained in details in the coming sections. The NFT faithfully represents the data; in other words for any member of the learning data set, it can correctly predict the corresponding class label. Fortunately construction of the decision tree using ID3 requires only this much information. We shall prove that one can exactly construct the decision tree from the NFT of the data. The link connecting the blocks at the lower-left corner of Figure 5 and the upper-right corner represent this route.

The direct link connecting the blocks corresponding to the "Dataset" and the "Decision tree" represents the approach that a traditional ID3 algorithm takes. The NFT and the Fourier spectrum of the tree are also

[1]Order of a coefficient $w_{\mathbf{j}}$ is the number of features defining the corresponding partition $\mathbf{j}$. For boolean features, the order is essentially the number of 1-s in the partition. Low-order coefficients are the ones for which the orders of the partitions are relatively small

| $\mathbf{x}$ | $f(\mathbf{x})$ | $\overline{f}(\mathbf{x})$ |
|---|---|---|
| 0 0 0 | 1 | 0 |
| 0 0 1 | 1 | 0 |
| 0 1 1 | 0 | 1 |
| 1 0 1 | 0 | 1 |
| 1 1 0 | 1 | 0 |
| 1 1 1 | 1 | 0 |

Table 1: Data vectors with their original and complemented function values

related. They share the same outcome for every member that is in the learning data set. We may be able convert either of them into the other; In fact, this may offer a viable way to construct the tree in a distributed fashion. This will be further explored in a later section. We may also be able to estimate the Fourier spectrum of the tree directly from the learning data. Although the Fourier spectrum of the tree is functionally equivalent to the tree itself, construction of the tree representation from the estimated Fourier coefficients is yet to be explored. The link between the two upper-right and the lower-right blocks represents this possible route.

Note that the NFT is not as same as the Fourier spectrum of the decision tree that an algorithm like ID3 may produce from the data. This is simply because the tree makes an inductive leap and it generalizes to the complete domain by assigning the class labels based on the information from the learning data set. This has very interesting implications that will be discussed later. This paper explores two of the possible approaches toward distributed construction of the tree through Fourier analysis. First, the route through construction of the NFT of the data is explored. This discussion is primarily of theoretical interest and it will help us understanding the subsequent practical approaches. Next, an alternate and possibly more practical approach through direct estimation of the Fourier spectrum of the tree is discussed.

## 4.5 From Non-uniform Fourier Coefficients to Entropy

| $j$ | $w_j$ | $w'_j$ |
|---|---|---|
| 000 | $0.\dot{6}$ | $0.\dot{3}$ |
| 001 | 0.0 | $-0.\dot{3}$ |
| 010 | 0.0 | 0.0 |
| 011 | 0.0 | 0.0 |
| 100 | 0.0 | 0.0 |
| 101 | 0.0 | 0.0 |
| 110 | $0.\dot{6}$ | $-0.\dot{3}$ |
| 111 | 0.0 | $0.\dot{3}$ |

| $\mathbf{h}$ | $\phi(\mathbf{h})$ | $\overline{\phi}(\mathbf{h})$ | (+/-) |
|---|---|---|---|
| 0 * * | $0.\dot{6}$ | $0.\dot{3}$ | 2/1 |
| 1 * * | $0.\dot{6}$ | $0.\dot{3}$ | 2/1 |
| * 0 * | $0.\dot{6}$ | $0.\dot{3}$ | 2/1 |
| * 1 * | $0.\dot{6}$ | $0.\dot{3}$ | 2/1 |
| * * 0 | $0.\dot{6}$ | 0.0 | 2/0 |
| * * 1 | $0.\dot{6}$ | $0.\dot{6}$ | 2/2 |

Table 2: (Left) Two sets of Fourier coefficients obtained from normal and complemented function values. (Right)Schemata of order one with their proportion-weighted fitness averages computed from normal/complemented function values.

As noted in an earlier section, construction of the decision tree using the ID3 algorithm requires computation of the entropy of different equivalence classes that we choose to call schema. This section shows that the schema entropies can be computed directly from the NFT. This section considers first the case where all the features are boolean. Following [4] let us define NFT in a rigorous fashion.

Let the proportion function $\mathbf{P}(\mathbf{x})$ be the number of instances of $\mathbf{x}$ divided by the size of the data set. Let us also define the *proportion-weighted class label*, $\phi(\mathbf{x})$ and its Fourier transformation as follows,

$$\phi(\mathbf{x}) = f(\mathbf{x})\mathbf{P}(\mathbf{x})2^l \qquad (7)$$

$$w_j = \frac{1}{2^l}\sum_{\mathbf{x}=0}^{2^l-1} \phi(\mathbf{x})\psi_j(\mathbf{x}) \qquad (8)$$

where $\psi_j(\mathbf{x})$ is the $j$-th basis function for Fourier transform. After finding the Fourier coefficients from the $\phi(\mathbf{x})$ values, we calculate the proportion of positive instances (class label 1) in a schema $\mathbf{h}$ as follows,

$$\phi(\mathbf{h}) = \sum_{\mathbf{j} \in \mathbf{J}(\mathbf{h})} w_j \psi_j(\beta(\mathbf{h})) \tag{9}$$

where,

$$J_i(\mathbf{h}) = \begin{cases} 0 & \text{if } h_i = *; \\ * & \text{if } h_i = 0, 1; \end{cases}$$

$$\beta_i(\mathbf{h}) = \begin{cases} 0 & \text{if } h_i = 0, *; \\ 1 & \text{if } h_i = 1; \end{cases}$$

Further details about NFT for binary features can be found elsewhere [4]. Let us illustrate these equations using an example. Consider Table 2(Left) that shows the Fourier Coefficients for $\phi(x)$. For example, $\phi(1**) = \sum_{\mathbf{j} \in \{000,100\}} w_j \psi_j(100) = w_{000} - w_{100} = 0.\dot{6}$. Note that $\phi(\mathbf{h})$ is nothing but the average of $\phi(\mathbf{x})$ values for all the members of schema $\mathbf{h}$, i.e.

$$\phi(\mathbf{h}) = \frac{1}{|\mathbf{h}|} \sum_{\mathbf{x} \in \mathbf{h}} \phi(\mathbf{x}) \tag{10}$$

where $|\mathbf{h}| = 2^{l - o(\mathbf{h})}$. Now using Equation 7 we get,

$$\sum_{\mathbf{x} \in \mathbf{h}} f(\mathbf{x}) = \frac{|\mathbf{h}| \phi(\mathbf{h})}{P(\mathbf{x}) 2^l} \tag{11}$$

Equation 11 represents the number of members in $\mathbf{h}$ with function value one (positive instance). Since we do not know the number of members of $\mathbf{h}$, in order to get the number of negative instances in $\mathbf{h}$, we need another set of non-uniform Fourier Coefficients that can be obtained from the same data set but with complemented $f(\mathbf{x})$ values. The rightmost column of Table 2(Right) shows the number of positive and negative instance vectors in each schema of order one. As a check, let us use Equation 11 to calculate the number of positive and negative instances of schemata $1**$ and $**0$. For the schema $1**$, number of positive and negative instances are obtained as,

$$\sum_{\mathbf{x} \in \{1**\}} f(\mathbf{x}) = \frac{|\mathbf{1}**| \phi(\mathbf{1}**)}{P(\mathbf{x}) 2^3} = \frac{4 \times 0.\dot{6}}{\frac{4}{3}} = 2$$

$$\sum_{\mathbf{x} \in \{1**\}} \bar{f}(\mathbf{x}) = \frac{|\mathbf{1}**| \bar{\phi}(\mathbf{1}**)}{P(\mathbf{x}) 2^3} = \frac{4 \times 0.\dot{3}}{\frac{4}{3}} = 1$$

$\bar{f}$ and $\bar{\phi}$ denote the complemented version of the functions $f$ and $\phi$ respectively. Now, for the schema $**0$ we can write,

$$\sum_{\mathbf{x} \in \{**0\}} f(\mathbf{x}) = \frac{|**\mathbf{0}| \phi(**\mathbf{0})}{P(\mathbf{x}) 2^3} = \frac{4 \times 0.\dot{6}}{\frac{4}{3}} = 2$$

$$\sum_{\mathbf{x} \in \{**0\}} \bar{f}(\mathbf{x}) = \frac{|**\mathbf{0}| \bar{\phi}(**\mathbf{0})}{P(\mathbf{x}) 2^3} = \frac{4 \times 0.0}{\frac{4}{3}} = 0$$

We get the exactly the same numbers as in Table 2(Right).

Since we can easily calculate the number of positive and negative instances of each schema, measuring information gain achieved by choosing an attribute is straightforward. We only need to give weights by dividing the number of instances in each schema by total number of instances in all schemata under consideration. For example, the weights assigned to $**0$ and $**1$ are $2/6 = 1/3$ and $4/6 = 2/3$ respectively. The resulting decision tree from the Fourier coefficients is identical to the one constructed by a regular ID3 approach and it is shown in figure 4. The following section considers the general case of building decision trees for non-binary features.

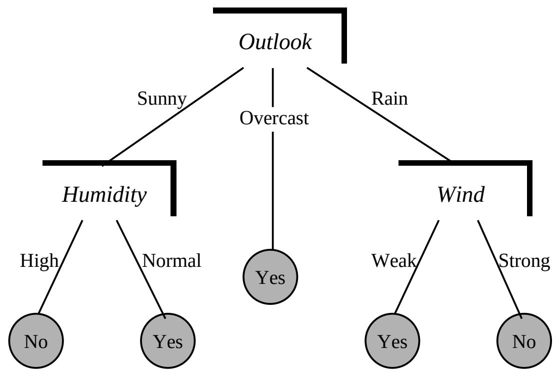


Figure 6: Non-binary feature decision tree.

## 4.6 Decision Trees With Non-binary Features

In data mining, we generally deal with data sets with non-binary features. Like *Outlook* in Figure 6, a feature may have 4cardinality(the number of attribute values) of three or more. With the same Fourier basis functions, the decision tree building algorithm developed in the previous section fails to work in these non-binary feature set. In this section, we extend the analysis by applying generalized nonuniform Fourier Transform in order to build a decision tree with non-binary features. The following analysis can handle $\overline{\lambda}$-ary features with different cardinality values.

Recall $\lambda$-ary Fourier basis functions over the set of features of the same cardinality $\lambda$ is defined as

$$\psi_{\mathbf{j}}^{(\lambda)}(\mathbf{x}) = \exp^{\frac{2\pi i}{\lambda}(\mathbf{x}\cdot\mathbf{j})} \tag{12}$$

The generalized $\overline{\lambda}$-ary Fourier function over data vector of length $l$ is defined as

$$\psi_{\mathbf{j}}^{\overline{\lambda}}(\mathbf{x}) = \Pi_{m=1}^{l} \exp^{\frac{2\pi i}{\lambda_m} x_m j_m} \tag{13}$$

where $\overline{\lambda} = \lambda_1, \lambda_2, ...\lambda_l$ and $\lambda_j$ denotes the cardinality of $j$-th feature($x_j$).

Similarly, we redefine proportion-weighted class label $\phi(\mathbf{x})$ and its $\overline{\lambda}$-ary Fourier transform as

$$\begin{aligned} \phi(\mathbf{x}) &= f(\mathbf{x})\mathbf{P}(\mathbf{x})\Pi_{i=1}^{l}\lambda_i & (14)\\ w_{\mathbf{j}} &= \Pi_{i=1}^{l}\frac{1}{\lambda_i}\sum_{\mathbf{x}}\overline{\psi}_{\mathbf{j}}^{\overline{\lambda}}(\mathbf{x})\phi(\mathbf{x}) & (15) \end{aligned}$$

where $\overline{\psi}_{\mathbf{j}}^{\overline{\lambda}}(\mathbf{x})$ is the complex conjugate of $\psi_{\mathbf{j}}^{\overline{\lambda}}(\mathbf{x})$.

Using the redefined $\phi(\mathbf{x})$, we can compute the average of *proportion-weighted class label*($\phi(\mathbf{h})$) as follows.

$$\phi(\mathbf{h}) = \sum_{l_1}\sum_{l_2}...\sum_{l_m} \exp^{2\pi i(\frac{l_1 b_1}{\lambda_{j_1}} + \frac{l_2 b_2}{\lambda_{j_2}} + ... + \frac{l_m b_m}{\lambda_{j_m}})} w_{(0,...,l_1,0,...,l_2,0,...,l_m,...0)} \tag{16}$$

where $\mathbf{h}$ has $m$ fixed bits $b_i$ at positions $j_i$ and $l_i$ has the cardinality of $\lambda_i$.

Table 4(Left) shows the proportion-weighted class label averages of three order one schemata and Table 4(Right) shows Fourier coefficients appeared in these average computations.

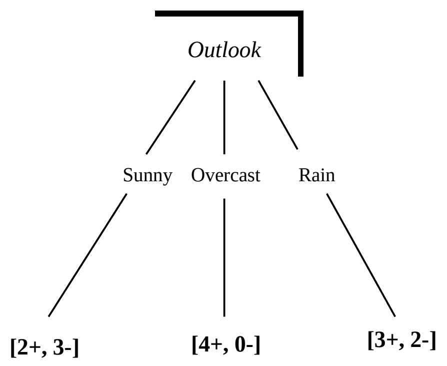


Figure 7: Positive/Negative instances split by choosing *Outlook*

| *Outlook* | *Temperature* | *Humidity* | *Wind* | *PlayTennis* |
|---|---|---|---|---|
| Sunny | Hot | High | Weak | No |
| Sunny | Hot | High | Strong | No |
| Overcast | Hot | High | Weak | Yes |
| Rain | Mild | High | Weak | Yes |
| Rain | Cool | Normal | Weak | Yes |
| Rain | Cool | Normal | Strong | No |
| Overcast | Cool | Normal | Strong | Yes |
| Sunny | Mild | High | Weak | No |
| Sunny | Cool | Normal | Weak | Yes |
| Rain | Mild | Normal | Weak | Yes |
| Sunny | Mild | Normal | Strong | Yes |
| Overcast | Mild | High | Strong | Yes |
| Overcast | Hot | Normal | Weak | Yes |
| Rain | Mild | High | Strong | No |

| $\mathbf{x}$ | $f(\mathbf{x})$ | $\overline{f}(\mathbf{x})$ |
|---|---|---|
| 2 2 1 0 | 0 | 1 |
| 2 2 1 1 | 0 | 1 |
| 1 2 1 0 | 1 | 0 |
| 0 1 0 0 | 1 | 0 |
| 0 0 0 0 | 1 | 0 |
| 0 0 0 1 | 0 | 1 |
| 1 0 0 1 | 1 | 0 |
| 2 1 1 0 | 0 | 1 |
| 2 0 0 0 | 1 | 0 |
| 0 1 0 0 | 1 | 0 |
| 2 1 0 1 | 1 | 0 |
| 1 1 1 1 | 1 | 0 |
| 1 2 0 0 | 1 | 0 |
| 0 1 1 1 | 0 | 1 |

Table 3: Data examples and their vector representations

For example, schema 0*** has average of 0.64286 which is obtained as,

$$\begin{aligned}
\phi(0***) &= \exp^{2\pi i \times \frac{0 \cdot 0}{3}} w_{0000} + \exp^{2\pi i \times \frac{0 \cdot 1}{3}} w_{1000} + \exp^{2\pi i \times \frac{0 \cdot 2}{3}} w_{2000} \\
&= w_{0000} + w_{1000} + w_{2000} \\
&= 0.64286 - 0.12372\mathbf{i} + 0.12372\mathbf{i} \\
&= 0.64286
\end{aligned}$$

To illustrate how we calculate the information gain using generalized($\overline{\lambda}$) discrete Fourier transform, consider the data set in Table 3 [28]. Notice that both attribute values and class labels are mapped to integers for $\overline{\lambda}$-ary Fourier transform. In Figure 6, *Outlook* of cardinality three is chosen as the root. The number of positive and negative instances split by choosing *Outlook* are shown in Figure 7. The schema representations of the paths along *Sunny*, *Overcast* and *Rain* links are 2***,1*** and 0*** respectively. The proportion-weighted class label averages of these schemata are found in Table 4. As we did in the previous section, we apply equation 11 to check the number of positive and negative instances along each schema. Notice that $\mathbf{P}(\mathbf{x}) = 1/14$ and

| $\mathbf{h}$ | $\phi(\mathbf{h})$ | $\overline{\phi}(\mathbf{h})$ | $j$ | $w_j$ | $w'_j$ |
|---|---|---|---|---|---|
| 0 * * * | 0.64286 | 0.42857 | 0000 | 0.64286 | 0.35714 |
| 1 * * * | 0.85714 | 0.0 | 1000 | -0.12372**i** | 0.03572 + 0.18557**i** |
| 2 * * * | 0.42857 | 0.64286 | 2000 | 0.12372**i** | 0.03572 - 0.18557**i** |

Table 4: Proportion-weighted class label averages of the first bit fixed schemata and some Fourier coefficients generated from data in Table 3

$\Pi_{i=0}^{l}\lambda_i = 3 \times 3 \times 2 \times 2 = 36$. For the schema 2***,

$$\begin{aligned}\sum_{\mathbf{x}\in\{2***\}} f(\mathbf{x}) &= \frac{|\mathbf{2}***|\phi(\mathbf{2}***)}{P(\mathbf{x})\Pi_l\lambda_l} = \frac{12*0.42587}{\frac{36}{14}} \doteq 2 \\ \sum_{\mathbf{x}\in\{2***\}} \overline{f}(\mathbf{x}) &= \frac{|\mathbf{2}***|\overline{\phi}(\mathbf{2}***)}{P(\mathbf{x})\Pi_l\lambda_l} = \frac{12*0.64286}{\frac{36}{14}} \doteq 3\end{aligned}$$

For the schema 1***,

$$\begin{aligned}\sum_{\mathbf{x}\in\{1***\}} f(\mathbf{x}) &= \frac{|\mathbf{1}***|\phi(\mathbf{1}***)}{P(\mathbf{x})\Pi_l\lambda_l} = \frac{12*0.85714}{\frac{36}{14}} \doteq 4 \\ \sum_{\mathbf{x}\in\{1***\}} \overline{f}(\mathbf{x}) &= \frac{|\mathbf{1}***|\overline{\phi}(\mathbf{1}***)}{P(\mathbf{x})\Pi_l\lambda_l} = \frac{12*0.0}{\frac{36}{14}} = 0\end{aligned}$$

And for the schema 0***,

$$\begin{aligned}\sum_{\mathbf{x}\in\{0***\}} f(\mathbf{x}) &= \frac{|\mathbf{0}***|\phi(\mathbf{0}***)}{P(\mathbf{x})\Pi_l\lambda_l} = \frac{12*0.64286}{\frac{36}{14}} \doteq 3 \\ \sum_{\mathbf{x}\in\{0***\}} \overline{f}(\mathbf{x}) &= \frac{|\mathbf{0}***|\overline{\phi}(\mathbf{0}***)}{P(\mathbf{x})\Pi_l\lambda_l} = \frac{12*0.42857}{\frac{36}{14}} \doteq 2\end{aligned}$$

Once again, we get the exactly same numbers as in Figure 7. As we see, a decision tree with non-binary features can also be constructed directly from the NFT spectrum of the data. However, the NFT of the data does not guarantee the nice properties like polynomial description and exponentially decaying magnitude (listed in section 4.3). The communication of the NFT of the data is not guaranteed to have low overhead. Therefore, we may need a technique to estimate the spectrum of the tree.

One possibility is to estimate the Fourier spectrum of the tree (not the NFT of the data) directly from the data. This approach has some strong advantages. Recall that the Fourier spectrum of a bounded-depth tree has some very favorable properties (refer to section 4.3). It has only a polynomial number of non-zero coefficients and moreover it can be approximated by only a small number of low-order coefficients. Therefore, the distributed computation of the tree can be very efficient using this approach. The following section presents this.

## 4.7 Building Decision Trees Through Direct Estimation of Its Fourier Spectrum

The Fourier representation of a bounded-depth decision tree is sparse and it can be approximated using only a small number of coefficients since the coefficients exponentially decay as the orders of the corresponding partitions increase. Therefore, if we can find a direct way to estimate these coefficients from data then construction of the global tree will require minimal communication. This section describes one approach to do that.

Note that the construction of an globally accurate tree requires only the low order coefficients of the spectrum. So we only need to estimate the coefficients for which the order is bounded by some small constant.

Let $\Xi$ be the set of all coefficient-indices and $\Xi_k$ be the set of those indices with order value less than equal to $k$. Let us divide the indices in $\Xi_k$ into two groups. Let $\Xi_k^{(l)}$ be the indices that can be computed at a data site using only the local feature values. Let $\Xi_k^{(c)}$ be the rest of the indices for whom we need features values from different sites. $\Xi_k = \Xi_k^{(l)} \cup \Xi_k^{(c)}$. Now we can write,

$$f(\mathbf{x}) = \sum_{i \in \Xi_k} w_i \psi_i(\mathbf{x}) + \sum_{j \in \Xi - \Xi_k} w_j \psi_j(\mathbf{x})$$

The error in approximating $f(\mathbf{x})$ using only the coefficients in $\Xi_k$ is,

$$\delta^{(t)}(\mathbf{x}) = \sum_{j \in \Xi - \Xi_k} w_j \psi_j(\mathbf{x}) = f(\mathbf{x}) - \sum_{i \in \Xi_k} w_i \psi_i(\mathbf{x})$$

This error term can be iteratively modeled using different sets of coefficients until the error converges. Let $S$ be the training data set and $|S|$ be the cardinality of this set. The proposed algorithm works in the following manner.

**Local site:**

- Initialize:
  - Set $\delta^{(t)}(\mathbf{x}) = f(\mathbf{x}), \forall x \in S; \delta^{(t)}(\mathbf{x}) = 0$ otherwise.
  - Set the coeficients $w_j^{(0)} = 0$ for all $j \in \Xi_k^{(l)}$.
- Iterate over $t$ until $\delta^{(t)}(\mathbf{x})$ continues to decrease toward the acceptable level of the approximation error:
  - Compute $w_j^{(t)} = \frac{1}{|S|} \sum_{\mathbf{x}} \delta^{(t)}(\mathbf{x}) \psi_j(\mathbf{x})$, such that $j \in \Xi_k^{(l)}$.
  - Set $\delta^{(t+1)}(\mathbf{x}) = \delta^{(t)}(\mathbf{x}) - \sum_{i \in \Xi_k} w_i^{(t)} \psi_i(\mathbf{x})$.
  - Update the estimate of the coefficients, $w_j^{(t+1)} = w_j^{(t-1)} + w_j^{(t)}$ for all $j \in \Xi_k^{(l)}$.

**Global site:**

- Collect all the locally computed coefficients and a small representative sample of the data from each sites.
- Set $\delta^{(t)}(\mathbf{x}) = f(\mathbf{x}) - \sum_{j \in \Xi_k^{(l)}, \forall l} w_j^{(t)} \psi_j(\mathbf{x})$.
- Iterate for all the coefficients corresponding to the cross-terms in $\Xi_k^{(c)}$. The iterative algorithm is exactly same as the one used for computing the local terms. The only difference is that $\Xi_k^{(l)}$ is replaced by $\Xi_k^{(c)}$. At each stage of the iteration, the error is computed over the sampled data set.

Note that the above calculation implicitly asserts $\delta^{(t)}(\mathbf{x}) = 0$ for all domain members that are not in the training data set while computing the Fourier representation of the error. This makes the computation of the coefficients possible by only considering the training data set. This approach to estimate the Fourier spectrum assumes that the tree has a bounded depth and therefore only the low order coefficients are required to approximate the tree. The algorithm then tries to approximate the function to be learned using only the low order coefficients defined by local features. The NFT assumes a function value of zero for every domain member that is not in the training data set. However, a decision tree constructed from the data does not do that. Instead the tree takes the inductive leap by assigning different function values to the member not in the training set based on the training data. The iterative process also does the same. It continues to approximate the contribution of the higher order coefficients using the partitions in $\Xi_k$ by inducing different and possibly non-zero function values for the domain members that are not in the training data set. Once the error stops decreasing, the local partitions cannot help any more. If the approximation error is significant then we need to compute the cross-terms. Since a bounded depth decision tree contains only a bounded number of cross-terms, estimation of the cross-terms from a relatively small sample set is feasible. As noted earlier in this paper, this estimation of the cross-terms will require $O(|\Xi_k^c|)$ data row communication from each of the sites.

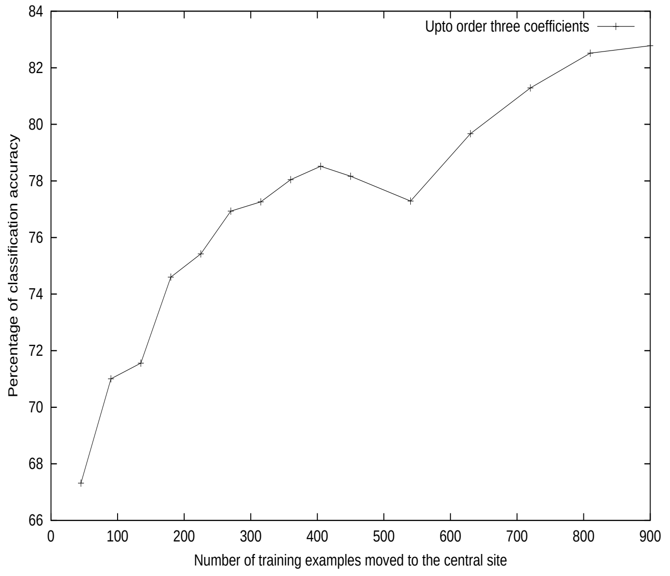


Figure 8: Variation of the classification accuracy with respect to the number of data points transferred to the central site for computing the cross-terms.

In order to show how the estimation accuracy varies with the size of the data sent to the central site, let us consider the experimental result presented in Figure 8. The set up considers a dataset of 20 boolean features, divided vertically and placed in two sites. Each site stores 104,857 training instances, which is about 10% of the complete domain. The classification problem is designed using a decision tree of bounded depth of seven, generated manually. Figure 8 shows the variation of the classification accuracy over a testing data set (similar in size to the training data set) with respect to the number of data points transferred to the central site for learning the cross-terms. As the experimental result shows, even with a small fraction of data instances, we could approximate the spectrum of the decision tree with good accuracy.

The portion of the work presented here is preliminary. The experiment considered all the coefficients up to order three. Since decision tree is a greedy search, a fair comparison will require incorporation of greedy search in the Fourier spectrum space. A hill-climbing technique is expected to reduce the number of higher order cross-terms considered in the central site. If the pruning is accurate then the cross-terms can be learned using even smaller number of examples. This will result in further reduction of communication overhead. An extended study of this technique is presented elsewhere [21]. The following section discusses the extension of the CDM for polynomial regression that has already been applied to large problems [15].

## 5 CDM and Regression

Regression [29], like decision tree learning, is a popular data modeling technique. As we noted earlier in Section 3, naive application of standard regression technique may produce misleading and ambiguous results in heterogeneous, distributed environment. Earlier, we also saw a simple CDM regression example for a toy problem with discrete binary-valued feature variables. In this section, we address the issue in details and present a method for distributed, polynomial-regression to the continuous-valued feature domain.

The CDM exploits the strength of a representation to describe a data model in a sparse and concise fashion. We saw that Fourier representation is good for representing decision trees. However Fourier or any such representation is unlikely to be good for every type of data model. Since there is no known technique to construct an appropriate technique in a domain independent fashion, the CDM research explores the suitability of such representations in a domain-specific fashion. In this section we choose Wavelet representation [16, 31, 30, 37, 38, 41] since Wavelet are widely acknowledged to have sparser representation in continuous domain with periodic patterns. Wavelets have already found many applications in signal processing, image

| $f(x_1, x_2)$ | $x_1$ | $x_2$ |
|---|---|---|
| $b_0 + b_1 + b_2 + b_3$ | 1 | 1 |
| $b_0 + b_1 - b_2 - b_3$ | 1 | $-1$ |
| $b_0 - b_1 + b_2 - b_3$ | $-1$ | 1 |
| $b_0 - b_1 - b_2 + b_3$ | $-1$ | $-1$ |

Table 5: Sample data for sparse representation example.

| Fourier Transform | | | | |
|---|---|---|---|---|
| 1 | $x_2$ | $x_1$ | $x_1x_2$ | $f(x_1, x_2)$ |
| 2 | 0 | 0 | 0 | $2b_0$ |
| 0 | 0 | $1+i$ | $1-i$ | $(b_1 + b_3) + (b_1 - b_3)i$ |
| 0 | 2 | 0 | 0 | $2b_2$ |
| 0 | 0 | $1-i$ | $1+i$ | $(b_1 + b_3) - (b_1 - b_3)i$ |
| **Wavelet-packet Transform** | | | | |
| 1 | $x_2$ | $x_1$ | $x_1x_2$ | $f(x_1, x_2)$ |
| 1 | 0 | 0 | 0 | $b_0$ |
| 0 | 0 | $-1$ | 0 | $-b_1$ |
| 0 | $-1$ | 0 | 0 | $-b_2$ |
| 0 | 0 | 0 | 1 | $b_3$ |

Table 6: Discrete Fourier transform vs. Wavelet-packet transform for spare representation example

compression, data analysis. A detailed description of these applications can be found elsewhere [41].

In order to further motivate the choice of Wavelet over Fourier consider the function $f(x_1, x_2) = b_0 + b_1x_1 + b_2x_2 + b_3x_1x_2$, and the associated data samples shown in Table 5.

If we apply a discrete Fourier transform and a Wavelet-packet (to be described in the coming section) transform to the data, we obtain the results presented in Table 6. The Wavelet transform is seen to provide a sparser representation of the feature variables, reflecting the orthogonal basis in the feature space. For this particular example, the Wavelet transform produces an orthogonal representation which is superior to that of the Fourier transform.

## 5.1 Wavelet Basis and Wavelet-Packet Analysis

A Wavelet basis consists of two sets of functions, scaling basis functions, $\phi_i$ and Wavelet basis functions, $\psi_i$, where the Wavelet functions are dilated and translated versions of the scaling functions [37, 38]. To understand the relation between the scaling and Wavelet functions, consider a vector space $S^j$ with $2^j$ dimensions defined on the interval $[0, 1)$. $S^j$ contains all functions defined on $[0, 1)$ which are piece-wise constant on $2^j$ equal sub-intervals. If $S^{j+1}$ is also defined on $[0, 1)$ then every function in $S^j$ is also in $S^{j+1}$ since each interval in $S^j$ may be considered to correspond to two contiguous intervals in $S^{j+1}$. Let $S^{j+1} = S^j + W^j$, where the subspace $W^j$ is the orthogonal complement of $S^j$ in $S^{j+1}$. If we assert that the basis functions for $S^j$ are the scaling functions, $\phi_i^j$, then the basis functions for $W^j$ will be the Wavelet functions, $\psi_i^j$. Note that since $S^j$ and $W^j$ are complementary orthogonal spaces, the $\phi_i^j$ and $\psi_i^j$ will be orthogonal to each other in $S^{j+1}$. If, in addition, the $\phi_i^j$ form an orthogonal basis for $S^j$ and the $\psi_i^j$ form and orthogonal basis for $W^j$, then combined the $\phi_i^j$ and $\psi_i^j$ form an orthogonal basis for $S^{j+1}$.

A simple set of scaling functions for $S^j$ are the scaled and translated "box" functions [37], defined on the interval $[0, 1)$ by:

$$\phi_i^j(x) = \phi(2^j x - i), i = 0, \ldots, 2^j - 1,$$

where

$$\phi(x) = \begin{cases} 1 & \text{for } 0 \le x < 1 \\ 0 & \text{otherwise} \end{cases}$$

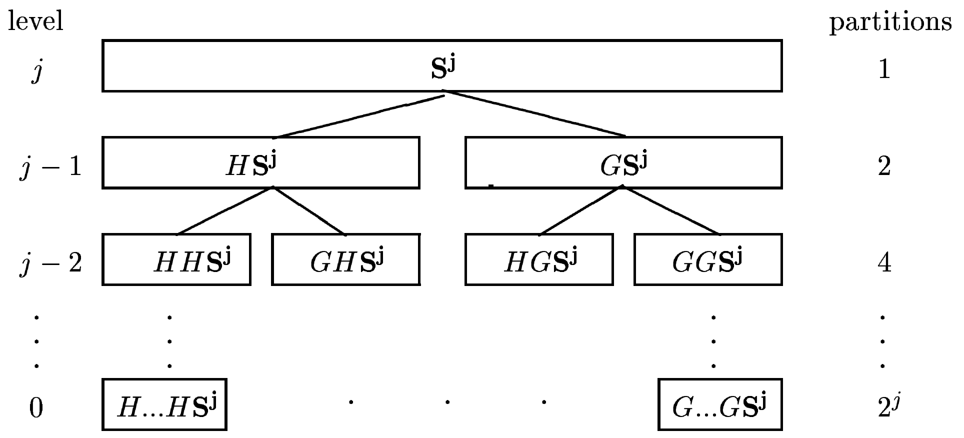


Figure 9: Application of quadrature filters in Wavelet-packet decomposition.

The Wavelet functions corresponding to the box basis functions are the *Haar Wavelets*:

$$\psi_i^j(x) = \psi(2^j x - i), i = 0, \ldots, 2^j - 1,$$

where

$$\psi(x) = \begin{cases} 1 & \text{for } 0 \le x < \frac{1}{2} \\ -1 & \text{for } \frac{1}{2} \le x < 1 \\ 0 & \text{otherwise} \end{cases}$$

Any function in $S^j$ may be represented in terms of these basis functions as

$$f(\mathbf{x}) = s_0^j \phi_0^j + s_1^j \phi_1^j + \ldots + s_{2^j-1}^j \phi_{2^j-1}^j = \mathbf{S}^j$$

or also as

$$f(\mathbf{x}) = s_0^{j-1}\phi_0^{j-1} + \ldots + s_{2^{j-1}-1}^{j-1}\phi_{2^{j-1}-1}^{j-1} + d_0^{j-1}\psi_0^{j-1} + \ldots + d_{2^{j-1}-1}^{j-1}\psi_{2^{j-1}-1}^{j-1}$$

The coefficients, $s_i^{j-1}$ and $d_i^{j-1}$ are generated by convolution of the $s_i^j$ with a set of orthogonal quadrature filters, $H$ and $G$. For the Haar Wavelets, $H = \{\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\}$ and $G = \{\frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}\}$.

The Wavelet-packet transform of a function in $S^j$ is calculated by recursively applying the quadrature filters to the $s$ and $d$ coefficients of the next lower dimension scale space and Wavelet space as if each represented a separate scale space. In this way, subspace coefficients are calculated for both the scale space and Wavelet space representing the next higher dimension scale and Wavelet spaces. Figure 9 shows how the quadrature filters are recursively applied to the scale and Wavelet subspaces in generating the Wavelet-packet transform. If the original function is in $S^j$ then $j$ recursive applications of $H$ and $G$ will result in $2^j$ orthogonal subspaces, $S_i^0, i = 0, \ldots, 2^j - 1$. At the top level, only a scale space exists and the $2^j$ function values are the coefficients for the "box" function basis for that space. Selecting the Haar Wavelets as the basis functions results in the coefficients of the $2^j$ orthogonal subspaces, $S_i^0$, representing a Walsh transform of the original function.

## 5.2 Polynomial Regression

One technique for implementing polynomial regression using Wavelet-based CDM is to first estimate locally the regression coefficients for terms which are based on feature subsets found in a single partition. Once these local coefficients are determined, they and a small subset of the sample data, are communicated to a central site to facilitate estimation of the coefficients for terms based on feature subsets which represent multiple partitions.

Given a partitioned set of real-valued features, $\mathbf{x}$, and a $k$-term polynomial function of those features, let $\Xi_A$ be the set of indices of terms which are functions only of features found in partition $A$. In partition $A$, we can form the terms $T_k(\mathbf{x}), k \in \Xi_A$ of the polynomial for each sample and apply the Wavelet-packet transform to the samples representing each term and to the samples of $f(\mathbf{x})$. Estimates of the local model coefficients, $a_k, k \in \Xi_A$, may be generated using standard regression techniques [29] directly on the Wavelet-packet transforms since $\mathbf{S}^0_{f(\mathbf{x})} \approx \sum_{k \in \Xi_A} a_k \mathbf{S}_{T_k}$ and the $\mathbf{S}_{T_k}$ are sparse, making them a nearly orthogonal basis for $\mathbf{S}_{f(\mathbf{x})}$. Once the coefficients of the local terms have been estimated, the coefficients of the terms containing cross-partition feature variables may be determined by communicating $O(m)$ samples, as described previously, and using standard system of linear equations or multiple regression techniques, depending on the sample data characteristics.

To demonstrate the CDM regression algorithm using this technique, we consider only the estimation of local model coefficients since it is through these coefficients that a majority of the information contained in the partitioned data set is communicated to the aggregate model. For this example, we consider a quadratic polynomial in 15 real-variables with six non-linear terms:

$$\begin{aligned} f(\mathbf{x}) \quad = \quad & 20x_0 + 5x_0^2 + 18x_1 - 8x_1^2 + 16x_2 + 13x_2^2 + \\ & 14x_3 + 11x_3^2 + 12x_4 - 14x_4^2 + 10x_5 - 8x_5^2 + \\ & 8x_6 + 11x_6^2 + 6x_7 + 13x_7^2 - 7x_8 - 12x_8^2 - \\ & 9x_9 + 15x_9^2 - 11x_{10} + 9x_{10}^2 - 13x_{11} - 10x_{11}^2 - \\ & 15x_{12} - 16x_{12}^2 - 17x_{13} - 10x_{13}^2 - 19x_{14} + 7x_{14}^2 + \\ & 5x_0x_1 - 3x_6x_{10} + 12x_4x_{11} - 8x_{12}x_{14} - 2x_{13}x_2^2 + 4x_4^2x_8^2 \end{aligned}$$

To observe the effect of sample set size, the CDM regression algorithm is applied to a series of data sample sets of increasing size. Each data sample set in the series contains $2^j$ data samples to allow for a simplified implementation of the Wavelet-packet transform algorithm. The data sample set series contains sets for $j = 6, \ldots, 15$. Wickerhauser [41] provides guidance on implementing the Wavelet-packet transform algorithm to handle the general case of sample size $\neq 2^j$. A single data sample consisted of 15 randomly generated values of $x_i, i = 0, \ldots, 14$, and the associated value of $f(\mathbf{x})$ determined by applying those $x_i$s to the polynomial. The random $x_i$ values were generated using the subtractive method given in [24]. The evaluation results are summarized in Figure 10 through Figure 14. Each plot shows the ratio of estimated to actual local term coefficient on the vertical axis and $log_2$ of the sample set size on the horizontal axis. The plots show that for each local term, as the sample set size increases from $2^6$ to $2^{15}$ the ratio of estimated to actual coefficient value converges toward 1.0. The results demonstrate that the Wavelet-packet transform based CDM regression algorithm produces accurate estimates of the model coefficients.

Another technique for implementing polynomial regression using Wavelet-based CDM is presented in [15]. This second technique differs from the method presented here in that instead of estimating local regression coefficients and communicating these and a sample data subset to a central site, the significant Wavelet coefficients for each feature are communicated to a central site. The coefficients of the terms in the polynominal may be estimated by performing the regression directly on the set of significant Wavelet coefficients representing those terms. Either technique may provide superior performance relative to the other in terms of higher accuracy and lower communication cost depending on the characteristics of the feature set to which it is applied.

The technique presented in [15] was evaluated using the same polynomial used here. Like the technique presented here, the second technique becomes more accurate as the number of samples in the data set increases. The accuracy of the second technique was also show to inversely depend on the number of terms in the polynomial based on feature subsets which represent multiple partitions and to depend on the compatablilty of the Wavelet basis selected with the data characteristics. Moreover, as the obtainable Wavelet representation of the data set becomes relatively more sparse, and the Wavelet coefficient values less uniform, the technique presented in [15] exhibits relatively better performance, all other factor being equal. In particular, time series data appears to exhibit the characteristics that result in better performance being realized using the second technique. Likewise, in cases where the Wavelet representation in relatively less spares with more uniform Wavelet coefficient values, the technique presented here performs relatively better.

Finally, [15] presents the results of an application of CDM regression to the problem of classification of the widely bench-marked Iris data [12]. Under the assumption that each of the four iris features resided in a separate partition, a three-fold cross validation test produced 90.3% accurate classification using a single Wavelet coefficient from each partition to build the aggregate classifier.

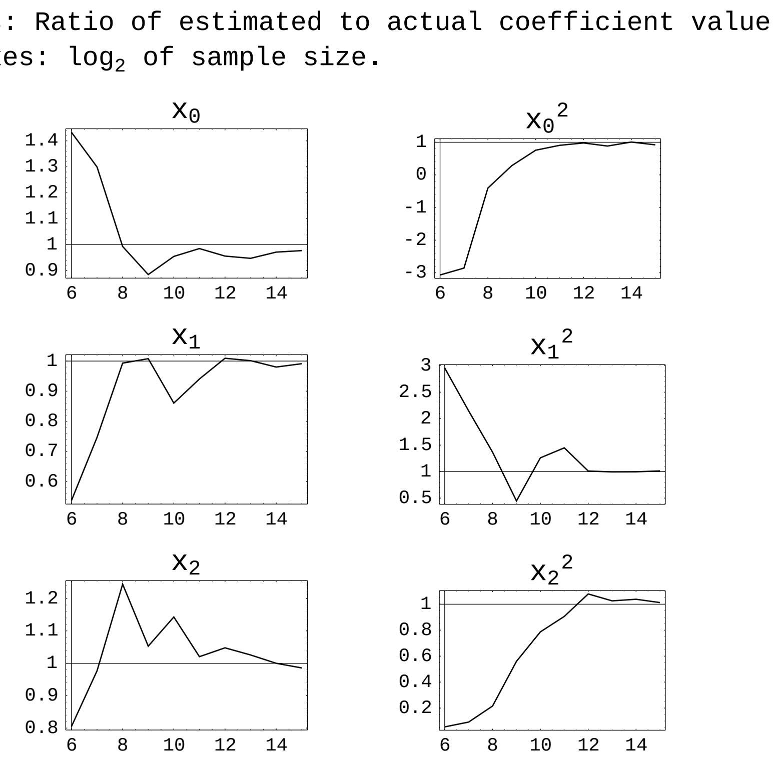


Figure 10: Ratio of Estimated to Actual Coefficient Value for Terms $x_0, x_0^2, x_1, x_1^2, x_2, x_2^2$ Converges Toward 1.0 as Sample Size Increases.

The following section describes an experimental system called BODHI for CDM-based distributed knowledge discovery.

## 6 BODHI: A Collective Data Mining-Based Experimental System

Application of the CDM to practical problems requires development of a system that employs CDM for distributed knowledge discovery. A successful DDM system should have the following characteristics:

**Communication Facilities:** The system must be able to communicate effectively between the various sites within the context of the system. Furthermore, the system should be built using existing protocols, and not concern itself with the underlying transport protocol (e.g., TCP/IP). The communication facilities of such a system must be able to handle the transfer of raw data, extracted knowledge, commands and command parameters to the learning algorithms, and even the learning algorithms themselves.

**Expandibility:** There are far too many different approaches and algorithms for machine learning to incorporate them all into a single system, and more are constantly being developed. Therefore, a CDM system must be able to incorporate new algorithms and methods as needed.

**Flexibility:** The system must be flexible enough to be adapted to different problems, while maintaining the expandibility capabilities noted previously, without significant changes to the core of the system.

y-axes: Ratio of estimated to actual coefficient value
x-axes: $\log_2$ of sample size.

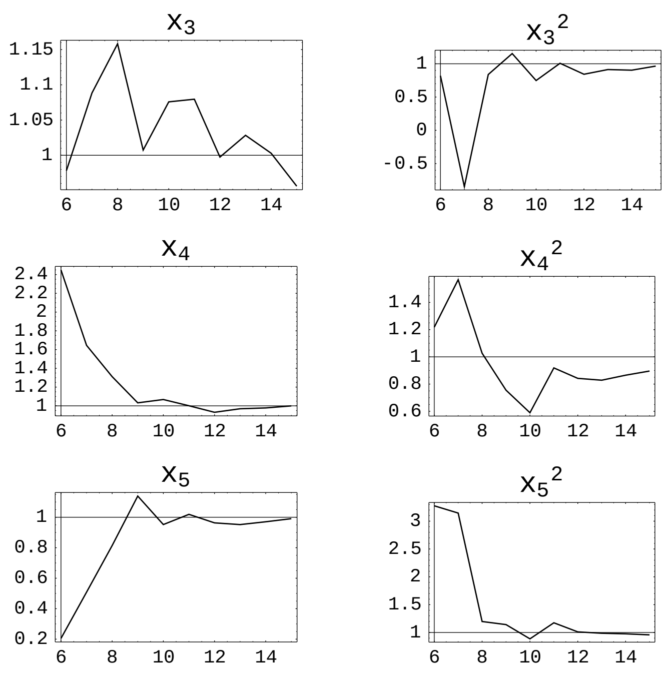


Figure 11: Ratio of Estimated to Actual Coefficient Value for Terms $x_3, x_3^2, x_4, x_4^2, x_5, x_5^2$ Converges Toward 1.0 as Sample Size Increases.

y-axes: Ratio of estimated to actual coefficient value
x-axes: $\log_2$ of sample size.

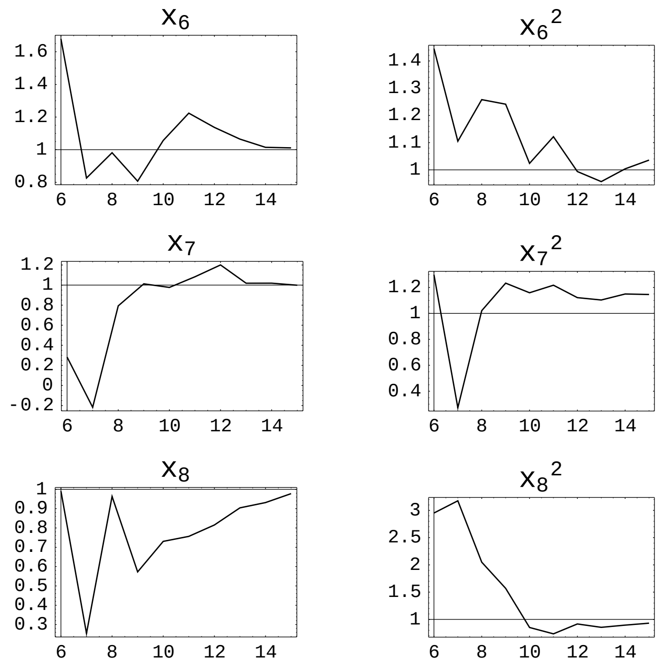


Figure 12: Ratio of Estimated to Actual Coefficient Value for Terms $x_6, x_6^2, x_7, x_7^2, x_8, x_8^2$ Converges Toward 1.0 as Sample Size Increases.

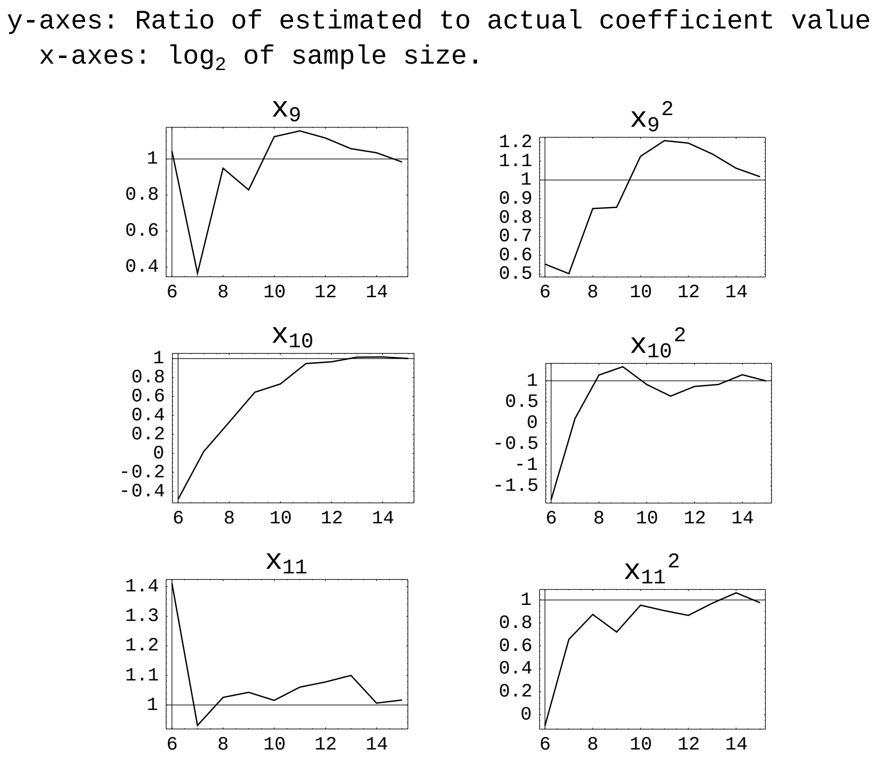


Figure 13: Ratio of Estimated to Actual Coefficient Value for Terms $x_9, x_9^2, x_{10}, x_{10}^2, x_{11}, x_{11}^2$ Converges Toward 1.0 as Sample Size Increases.

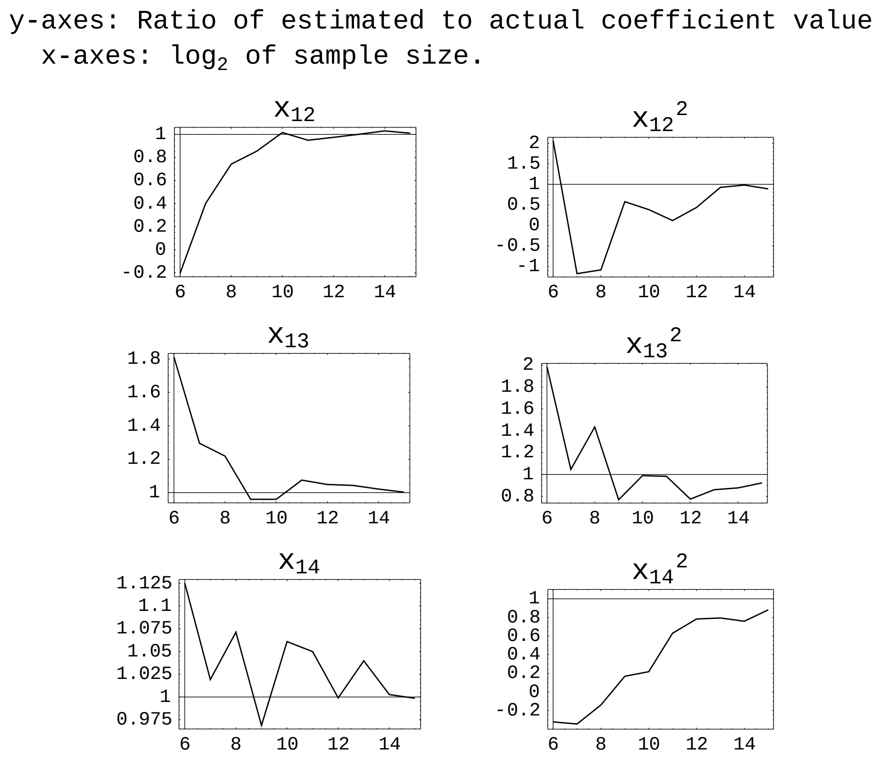


Figure 14: Ratio of Estimated to Actual Coefficient Value for Terms $x_{12}, x_{12}^2, x_{13}, x_{13}^2, x_{14}, x_{14}^2$ Converges Toward 1.0 as Sample Size Increases.

**Mobility:** It is important that when portions of the mining process must be performed in a sequential manner, as opposed to a parallel manner, that the CDM system be capable of allowing an algorithm to start at one location and then continue at another location.

**Application Independent Representation of Data Models:** An effective CDM system must be able to convey information regarding the partial data models among the various sites in the environment. This representation should not be system-specific. In other words, various sites must be able to communicate their knowledge in an agreed upon format regardless of the learning algorithm being applied to the problem.

**Platform Independence:** For such a system to function well within the context of a heterogeneous mixture of platforms and operating systems, the system should be made as platform independent as possible.

**Security:** The issue of security should be addressed. There are three primary areas that need to be addressed: authentication, encryption, and access permissions for the system.

**Centralized and Distributed Control** : The system should be able to handle both centralized and distributed control. Implementing centralized control is relatively straight-forward. This mode will be useful when the users of the system are centrally located. On the other hand, a distributed control mechanism will be needed to support collaborative, distributed user-interaction.

The BODHI (Beseizing knOwledge through Distributed Heterogeneous Induction) System, currently under development, addresses all of the above issues. Although additional facets of DDM systems are likely to emerge in the future, we believe the characteristics, listed above, offer a good starting point for developing a DDM system. For example, the BODHI system was first reported in [20], and since that point in time it has undergone significant design changes. Some of these changes came from practical issues that arose during the implementation process, and others out of further study and reflection upon the problems being addressed.

## 6.1 Design Principles

The BODHI System is an agent-based, distributed knowledge discovery system, that offers a transparent running environment and message exchange system, capable of handling mobile agents. The primary design goal of the BODHI system was to create a communication system and run time environment for use in collective data mining which was not bound to any specific platform, learning algorithm, or representation of knowledge. It was considered critical not to put limitations upon the system that would prevent its use in a variety of contexts and with a variety of distributed learning applications. Therefore, the BODHI system was built as an extensible system, providing a framework for communication between, and control of, distributed agents within the system, but without being bound to any specific platform, knowledge representation or machine learning algorithms. To prevent the limitation of the platforms upon which the system could be used, the core of the system was developed using Java. In order to prevent limitations being placed upon the use of the system concerning the learning algorithms utilized or the knowledge and data format, the system provides a generic, extensible framework that is easily adapted to any number of learning algorithms and knowledge and data formats. It should be noted that the BODHI system operates independently of the learning algorithms implemented in the machine learning agents which operate under its control.

The BODHI system is a modularized system, designed using object oriented principles. There is a precise division of responsibility for various tasks between the various components of the system. The primary component of the system is the *facilitator* module, which is responsible for directing data and control flow between the various distributed sites and the interfaces. Each local site has a communication module, known as the *agent station*, which is responsible for providing communication between the given local site and other sites, in addition to providing the runtime environment for the *agents*. Furthermore, the agent stations are responsible for communication security. The agent object within the BODHI framework is an extensible Java object that is used as the interface between the user implemented learning algorithm. This learning algorithm may be implemented either as an extended agent using Java only or as native code on the local machine. The individual agents and their learning algorithms are intended to be autonomous, and to this end, are not provided as a portion of the BODHI system. In the following discussion, the extensible agent class will be referred to simply as the agent class, or simply as an agent, and an agent class that has been extended to perform a specific learning task will be referred to as a learning agent. Finally, the *user interface* forms the final component of this system, and allows the user of the system to control the individual agents and schedule the events occurring within the context of the system as necessary.

The BODHI system is designed as an extensible hierarchy. Forming the base of the hierarchy are the individual agents. These agents are the components within the system that, when extended to specific learning algorithms, perform the actual machine learning for the given task. There may be any number of agents operating on a given site with in the system, or no agents, on a given site within the system. Agents within the system are autonomous, with minimal restraints placed upon their implementation. These restrictions and criteria are discussed below, but in general, there are two restrictions: (1) that the agents within a given instantiation of the system be able to communicate using a user defined data and knowledge format, and (2) that certain control functions be implemented in each extended learning agent.

The agent station forms the second layer of the hierarchy. Each distributed site within the system is required to have a single agent station. The agent station on a given site is responsible for starting and stopping the agents operating on its site, in addition to keeping track of the capabilities of the agents upon that site. Furthermore, the agent station is responsible for providing the run time environment for all agents operating under its control, in addition to routing all incoming and outgoing communication from and to agents operating on that site.

The third layer of the hierarchy consists of the facilitator. The facilitator itself, for purposes of ease of communication, is actually an extended agent; however, it should be noted that from a design perspective, the facilitator is higher in the hierarchy than any agent that has been extended to perform a specific machine learning task. The facilitator is responsible, in conjunction with the agent stations, for coordination of communication security. Furthermore, the facilitator is responsible for tracking of access permissions for the individual agents and agent types.

All communication between the individual agents is accomplished using a message subsystem built into the BODHI system. This generic messaging system allows communication between the individual agents, agent stations, the facilitator, and the user interface. This messaging system is based upon a generic *message* class, which was loosely based upon the KQML format [11], and is intended to allow new types and formats of messages to be added into the system as necessary. A message may carry either data or a command, or some combination thereof.

The message structure within the BODHI system was designed to be as generic as possible, so as not to constrain future expansions of the system, or limit the capabilities of the system through unnecessary constraints being placed upon the representation of the knowledge or data. Therefore, it is the responsibility of the individual agent to be aware of any message types that the given agent may encounter. Furthermore, when a new agent is added to the system, the knowledge and data formats understood by the new agent are registered in the facilitator.

Two types of security issues arose during the design process for the BODHI system: security of transmissions and access control of the various individual agents. Security of transmissions between two different sites was considered to be a critical point in the design of the system. In particular, two concerns arose: (1) that outsiders might be able to insert commands or data into the flow of messages within the system, and (2) that outsiders might be able to intercept data when it was transmitted from one site to another. These issues are being addressed through the current ongoing implementation of an RSA [35] based encryption scheme. The issue of access control for the individual agents is currently being addressed through the development of a system of permissions, based both upon permissions for an individual agent and permissions based upon types of agents and the "domain" within which they are operating.

The BODHI system was designed to perform certain specific tasks, including control related tasks, such as initialization, shutdown, and movement of agents, and data flow related tasks, such as the transfer of raw data and extracted knowledge between the individual agents and the end user. The minimal set of basic functionality provided within the BODHI system is listed below:

**Initialization and Shutdown:** One of the primary tasks of the system is to initialize and shut down the system on trusted remote sites.

**Agent Control:** The BODHI system provides the basic framework for passing of control sequences to and between the agents and agent stations within the system. While it is required that a set of basic commands be incorporated into all agent types that are to be incorporated into the system, the specific implementation of these commands is left to the user of the system. Of course, additional commands may be added to individual agent implementations as needed.

**Agent Mobility:** In many cases, it is necessary for an agent to act as a mobile agent. As all agents are extensions of a basic agent object, the BODHI system is easily capable of transferring an agent from one site to another, along with the agent's environment, configuration, current state, and learned knowledge.

**Transmission of Information:** The BODHI system provides the basic functionality for the agents to communicate information, both knowledge and data, to other agents within the system. As with the control flow capabilities provided by the BODHI system, many of the specifics of the type and format of the information and data are left to the end user. To wit, the framework for passing data is provided by the message subsystem, but the type and content are fields within the messages themselves which may be defined by the end user.

**User Interface:** The BODHI system is designed to allow the distributed mining process to be controlled through a user interface that resides upon the primary, initializing node within the system. Therefore, the BODHI system provides the framework for the user interface to monitor and control the agents within the system. While the specific interfaces for specific agents are not provided, when an agent is extended to perform a specific machine learning task, the user interface must be likewise extended.

## 6.2 System Components

The BODHI system consists of five primary component types: (1) the individual agents, which are autonomous entities which perform specific learning tasks; (2) The agent stations, which are responsible for providing the run time environment and for communication between the agents and other sites within the system; (3) The facilitator, which is responsible for coordinating communication between the various agent stations; (4) The user interface, by which the user of the system is able to configure and control the system, and (5) The individual messages which are passed through the system. Each of these is described below.

### 6.2.1 Agents

Agent is the extensible base class for any specific learning algorithm to be incorporated into the BODHI system. The BODHI system does not provide any specific learning algorithms bound to the agent class. Rather, the basis for using the agent class as the interface between the actual learning agent and the rest of the BODHI system is provided. It should be noted that the actual machine learning algorithm my be implemented as a Java extension to the agent class, or by extending the agent class in such a manner as to cause it to call native code on the host machine for the agent.

Agents in the BODHI system are mobile entities; that is, they are capable of being transferred from one site to another. When an agent moves from one site to another, all necessary information for the agent, including its current state, store of acquired knowledge, environment, and configuration, is moved with the agent.

There are seven methods which reflect the various states of the life cycle of an agent. These methods must be extended for each particular type of learning agent to be used within the system. While this is not intended to be a manual on the use of the BODHI system, it is informative to examine the basis for these seven methods that reflect the life cycle of the agents:

**init:** The *init* method initializes the agent. All necessary initializations for the configuration of the agent are performed using this method.

**start:** The *start* method initiates the actual main learning algorithm that the agent is to perform.

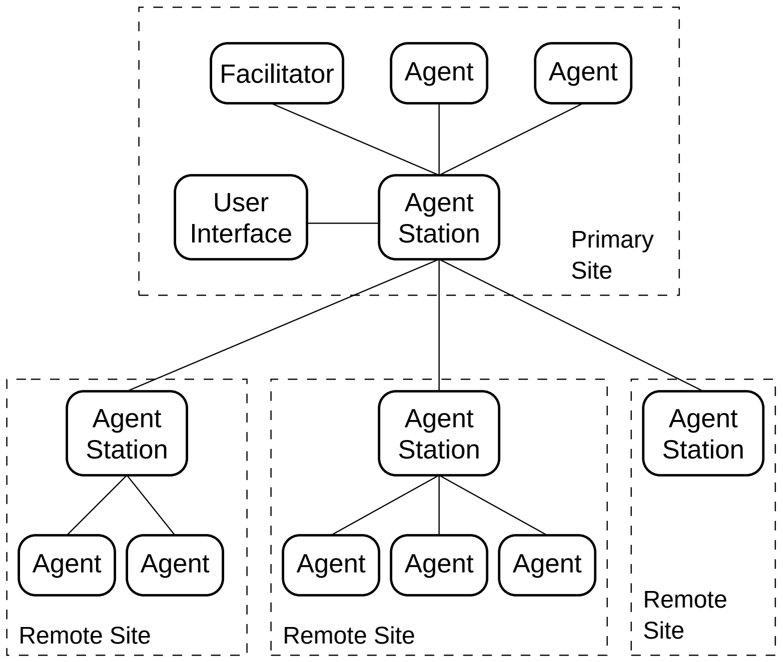


Figure 15: *Overall Systems Diagram for the BODHI system.*

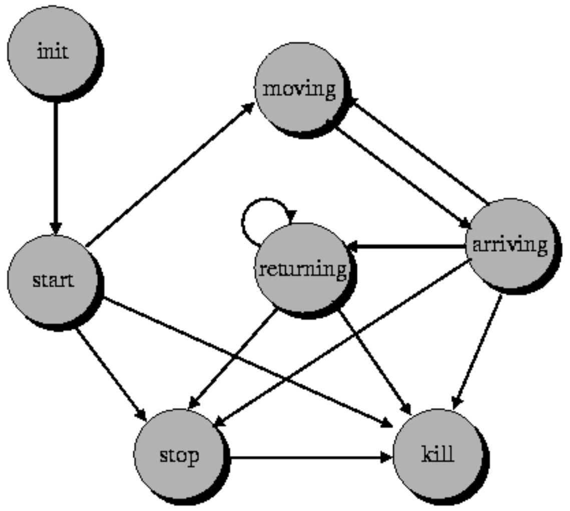


Figure 16: *Life cycle state diagram for an agent within the BODHI system.*

**moving:** The *moving* method is used to prepare the agent to be moved from one site to another. When an agent is to be moved, all configuration information, acquired knowledge, environment information and current state must be bundled with the agent. Once this has been accomplished, the agent's agent station transmits the agent as a message. It is necessary once the agent has moved to another site to call the *arriving* method. The agent is required to maintain a list of sites that have been visited.

**arriving:** The *arriving* method is the method used to restore an agent after it has been moved from one site to another. This method will restore the state of the agent at the new site, and will then cause the agent to continue in its learning process from where it left off at the point that the moving call was made.

**returning:** The *returning* method is similar to the *moving* method; however, in this case, the agent is prepared to return to the site from whence it originally came.

**stop:** The *stop* method stops the execution of the learning algorithm, and finalizes the results of the learning process. Once the any necessary cleanup has been performed, the *kill* method is called.

**kill:** The *kill* method halts the execution of the agent, and removes the agent from the system without any finalization processes being called.

Each agent is also associated with its own message queue. It is the responsibility of the individual agent to check the status of the queue periodically, and to respond to the messages within the queue as is appropriate. Certain control messages arriving at an agent's site (via the agent station) concerning that agent may be trapped and acted upon by the agent station (see below). It is the responsibility of the agent station to check if the arriving message is of this type (e.g., a kill message). If the arriving message does not fall into this category, it is simply placed in the agent's message queue, and becomes the agents responsibility to act on as needed.

Each individual agent and agent type is associated with certain permissions which control upon which sites and in which domains an agent may be instantiated, and what sort of movement is allowed. Furthermore, agents and agent types are associated with certain file system privileges. For more information, see the following section concerning security.

### 6.2.2 Agent Stations

Each node within the system contains a single instance of an agent station. The agent station is a daemon process responsible for providing the runtime environment for the agents, and passing messages between the individual agents, other agent stations, and facilitator. In addition, all encryption and decryption of all messages is performed by the agent station. The individual agents are never permitted direct access to any of the encryption and decryption routines.

The agent station at a given node is responsible for a number of specific tasks. It is the responsibility of the agent station to receive messages passed from other nodes within the system, and either take action upon them immediately or to pass them to the appropriate agent under the control of the agent station via that agent's message queue. It is also the responsibility of the agent station to initialize the individual agents when they are called into existence, and to clean up after the agents.

When a message is received at a given agent station, through a special message listener, that message is passed to the message handler. If the received message is an instruction to create a new instance of an agent, the message handler passes the message to an agent loader, which is then responsible for creating the new instance of the agent from either the network or from the local file system. Otherwise, the message is passed to the message queue of the individual agent, and it becomes the responsibility of the agent to act upon it.

Prior to loading a new agent, the agent loader will verify that the request for the generation of the new instance of an agent is permissible and possible. A special configuration class associated with the agent is used for this verification that lists all of the necessary components of the agent to be generated, such as necessary classes and other external files. Furthermore, the representation and knowledge type is verified to be the same as the other agents operating under the system to ensure compatibility between the instantiated agents.

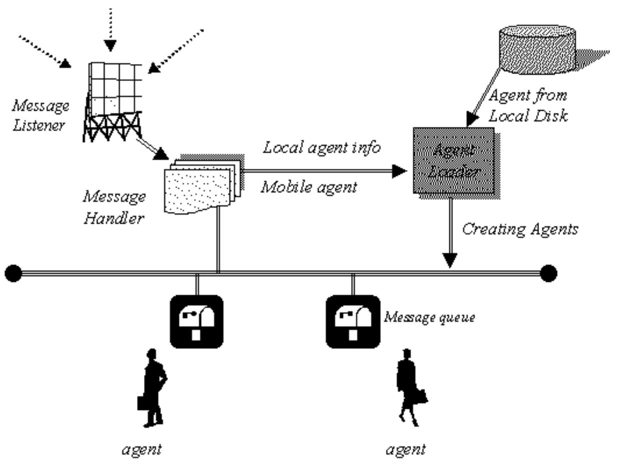


Figure 17: *Communication flow within a BODHI Agent Station.*

### 6.2.3 The Facilitator

The facilitator for the system is itself a special purpose agent. However, it is not intended to perform a machine learning task; rather, the job of the facilitator is to coordinate the communication and control flow between the individual agents. The facilitator is also responsible routing messages between the agents, and tracking the location of the individual agents.

The facilitator, like any other derived agent, operates under an agent station. Therefore, as with any other agent type, the facilitator never is to be allowed direct access to the encryption and decryption routines. However, unlike any other agent type, the facilitator is not permitted to move from one site to another; it must remain upon at the site where it originated.

The primary task of the facilitator is to coordinate communication between the agents. To this end, the facilitator is able to respond appropriately to several calls similar to the KQML scheme: ask one, ask all, tell one, and tell all. These functions allow a message routed through the facilitator to be sent to the appropriate destination or destinations.

The facilitator is responsible for resolution of the type and content of the knowledge represented in the messages passed between agents. When such resolution is necessary, the request will be passed up from the agent through the agent station and to the facilitator, which will then pass the response back to the agent.

Finally, the facilitator is responsible for pre-scheduled coordination of tasks within the overall system by the various agents and agent stations. To this end, when such a scheduling is necessary, the facilitator class is extended to include this functionality. As with much of the other adaptive functionality within this system, the framework for the functionality is provided, but it is left to the end user to implement the specific algorithms.

### 6.2.4 User Interface

The user interface for the BODHI system is designed, as are the other components of the system, to be expandable and adaptable to the types of agents that the user may incorporate into the system. Therefore, the base user interface is not intended to be the final user interface, but rather, a general basis for controlling the system.

The user interface of the BODHI system runs on the same machine as the facilitator, and communicates through the facilitator to the other components within the system. The user interface process cannot move from the machine that it was started upon.

### 6.2.5 Messages

Messages are the final component of the BODHI system. All communication between the various components, including all agents, the user interface and the facilitator, of the BODHI system is accomplished using messages.

Once the security subsystem is implemented, all messages are to be encrypted by the agent station at the sending machine, and are to be decrypted by the receiving machine's agent station. Furthermore, every message will be signed so as to allow verification of the origin of the message.

In order to preserve the ability for the system to handle any data, knowledge or control strings needed for a given use of the system, the message format is defined in such a manner as to allow it to carry virtually any kind of content. The message consists of four primary sections: the envelope, the message type, the content description, and the message content.

The envelope portion of the message is the addressing portion of the message. This allows the sender and receiver, in addition to the node (agent station) upon which the sender and receiver are operating, to be identified. The envelope portion of the message contains four primary entries: the sender, the receiver, the sending agent station, and the receiving agent station. It should be noted that it is the responsibility of the facilitator to route the message; when the sender sends the message, it does not need to know the actual, real location (i.e., IP address) of the recipient; it need only know the system-assigned name of the recipient.

The message type determines what the purpose of the message is. The message type can be divided into three types, which are further divided into sub parts. These three primary types are command, ask, and tell. The command type of message passes a command to either an agent station, telling that station to perform some action, such as creating an agent, or to an individual agent. The ask type is a request for information from a specific agent, and may be either a unicast or broadcast request for information. The tell type is an agents way of sending data, and also can be of a unicast or broadcast type.

The content description of a message is a code signifying what the content of the message contains, and allows the recipient to parse the content of the message. It is the responsibility of the facilitator to maintain the table of types of messages, and it is further the responsibility of the specific agent to be able to respond to the message in an appropriate manner.

Finally, the content portion of the message is the main body of the message. Its encoding is determined by the content description field of the message. It is the responsibility of the individual agent to be able to understand this section based upon what the content description of the message indicates the format of the content section to be.

## 6.3 Security Issues

There are two types of security issues relating to the BODHI system. The first is message security, and involves outsiders being able to impersonate or intercept transmissions. The second issue involves user (or agent) privileges in terms of locations upon which an agent may operate, and what data it is able to access as it moves from one site to another.

### 6.3.1 Message Security

There were two primary concerns relating to message security during the design. The first concern was that outsiders might be able to insert a command or data message into the system. The second concern was that an outsider might be able to intercept data when it was transmitted from one node to another. It should be noted that at a given site, the BODHI portion of the system operates as a single threaded process, and therefore, there is no danger of interception of messages between an agent and its agent station.

The description of the security portion of the BODHI system is based upon the design performed. The actual implementation of the security system is still underway.

The approach to message security to be taken within the BODHI system involves two phases. First, the initial transmission that creates a new agent station on a remote machine is to be encrypted using an RSA public key encryption scheme. This requires that the remote machine have a private key associated with it, and that the initial node have the public key for the remote machine.

Included as an argument to the (encrypted) initialization command to the agent station process at the remote machine will be a private key. The associated public key will be broadcast to the other agent stations within the system. All future transmissions to the new agent station will be signed by the sender, and encrypted using the public key. This will ensure that false messages cannot be inserted into the system, and, furthermore, that the messages cannot be intercepted by an outside party.

### 6.3.2 Access Control

The second security issue relating to this sort of system involves access control, and control permissions. It is necessary that the data available to a given agent be controlled, especially when an agent may move from site to site. Furthermore, it is necessary that, as agents are allowed to send control commands to one another, that there be a method for controlling permissions for agents issuing commands to one another.

Access control is performed by associating a list of permissions with each individual agent and agent type. The facilitator is responsible for keeping track of these permissions. These permissions include the listing of which site an agent is allowed to operate upon, what sort of agent movement is permissible, and what files are available upon the local file systems to the specific agent and type of agent. There is a global configuration for each type of agent, listing these parameters. When instance of an agent is instantiated, these permissions may be modified. It should be noted that this modification may include a further limitation of the permissions, but never an increase in the available permissions. The following section concludes this paper.

## 7 Conclusions And Future Work

Heterogeneous data sites are common in business, government, defense, and scientific information processing environments. The field of DDM must develop a well-grounded approach to deal with this general situation. Without that DDM is unlikely to be a viable alternative to current centralized data mining systems. The CDM technology offers one possible approach to this. The material presented in this paper is a result of our preliminary investigation and the framework is currently further explored and extended [15, 17]. However, the preliminary results demonstrate that CDM may offer a well grounded methodology for generating accurate global data models in a distributed fashion.

The CDM notes that almost all the practical data models have polynomial description. As a result it is often possible to find a polynomial description of a data model in some appropriate representation. If we can do that, we should also be able to build the model in a distributed fashion with only polynomially bounded amount of communication. As we see, the success of the CDM depends upon finding an appropriate representation and designing efficient algorithms to compute such representation. In this paper we have seen that this may be possible for polynomial regression and decision trees. We need to investigate this approach for other types of data models like Bayesian, Hidden Markov Models, and Neural networks. We are also extending the CDM work to the domain of unsupervised learning algorithms like hierarchical clustering [17]. Additional work on efficient construction of orthonormal representations, BODHI system development, multi-media DDM, and various applications is ongoing.

## Acknowledgements

The authors would like to acknowledge support from the National Science Foundation Grant IIS-9803360 and the American Cancer Society.

## References

[1] J.M. Aronis, V. Kolluri, F.M. Provost, and B.G. Buchanan. The WoRLD: Knowledge discovery from multiple distributed databases. In *Proceedings of Florida Artificial Intelligence Research Symposium (FLAIRS-97)*, page Not available, 1997.

[2] L. Breiman. Bagging predictors. *Machine Learning*, 26(2):123–140, 1996.

[3] L. Breiman. Combining predictors. In *Combining Artificial Neural Nets*, pages 31–50. Springer-Verlag, 1999.

[4] C. L. Bridges and D. E. Goldberg. The nonuniform Walsh-schema transform. In G. J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 13–22. Morgan Kaufmann, San Mateo, CA, 1991.

[5] P. Chan and S. Stolfo. Experiments on multistrategy learning by meta-learning. In *Proceeding of the Second International Conference on Information Knowledge Management*, pages 314–323, 1993.

[6] P. Chan and S. Stolfo. Toward parallel and distributed learning by meta-learning. In *In Working Notes AAAI Work. Knowledge Discovery in Databases*, pages 227–240. AAAI, 1993.

[7] P. Chan and S. Stolfo. Toward scalable learning with non-uniform class and cost distribution: A case study in credit card fraud detection. In *Proceeding of the Fourth International Conference on Knowledge Discovery and Data Mining*, page o. AAAI Press, September 1998.

[8] D. Cheung, V. Ng, A. Fu, and Y. Fu. Efficient mining of association rules in distributed databases. *IEEE Transaction on Knowledge and Data Engineering*, 8(6):911–922, 1996.

[9] V. Cho and B Wüthrich. Toward real time discovery from distributed information sources. In Xingdong Wu, Ramamohanarao Kotagiri, and Kevin B. Korb, editors, *Research and Development in Knowledge Discovery and Data Mining*, number 1394 in Lecture Notes in Computer Science : Lecture Notes in Artificial Intelligence, pages 376–377, New York, 1998. Springer-Verlag. Second Pacific-Asia Conference,PAKKD-98, Melbourne, Australia , April 1998.

[10] I. Dhillon and D. Modha. A data-clustering algorithm on distributed memory multiprocessors. Proceedings of the KDD'99 Workshop on High Performance Knowledge Discovery, 1999.

[11] T. Finin, Y. Labrou, and J Mayfield. KQML as an agent communication language. In J. Bradshaw, editor, *Software Agents*, pages 291–316. MIT Press, 1997.

[12] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Anals of Eugenics*, 7:179–188, 1936.

[13] R. et al. Grossman. The preliminary design of papyrus: A system for high performance, distributed data mining over clusters, meta-clusters, and super-clusters. In *Advances in Distributed and Parallel Knowledge Discovery*, page Not available. AAAI/MIT Press, 1999.

[14] Y. Guo and J. Sutiwaraphun. Knowledge probing in distributed data mining. In *Advances in Distributed and Parallel Knowledge Discovery*, page Not available. AAAI/MIT Press, 1999.

[15] D. Hershberger and H. Kargupta. Distributed multivariate regression using wavelet-based collective data mining. Technical Report EECS-99-02, School of EECS, Washington State University, 1999.

[16] Barbara Burke Hubbard. *The World Accoring to Wavelets*. A. K. Peters, Ltd., Wellesley, MA, 1998.

[17] E. Johnson and H. Kargupta. Collective, hierarchical clustering from distributed, heterogeneous data. To be published in the Lecture Notes in Computer Science, volume 1759, Springer-Verlag, 1999.

[18] H. Kargupta, I. Hamzaoglu, and B. Stafford. Scalable, distributed data mining using an agent based architecture. In David Heckerman, Heikki Mannila, Daryl Pregibon, and Ramasamy Uthurusamy, editors, *Proceedings of Knowledge Discovery And Data Mining*, pages 211–214, Menlo Park, CA, 1997. AAAI Press.

[19] H. Kargupta, I. Hamzaoglu, B. Stafford, V. Hanagandi, and K. Buescher. PADMA: Parallel data mining agent for scalable text classification. In *Proceedings Conference on High Performance Computing '97*, pages 290–295. The Society for Computer Simulation International, 1996.

[20] H. Kargupta, E. Johnson, E. Riva Sanseverino, H. Park, L. D. Silvestre, and D. Hershberger. Scalable data mining from distributed, heterogeneous data, using collective learning and gene expression based genetic algorithms. Technical Report EECS-98-001, School of Electrical Engineering and Computer Science, Washington State University, 1998.

[21] H. Kargupta and B. Park. Large scale distributed learning of decision trees from heterogeneous data. In preperation, November 1999.

[22] H. Kargupta and H. Park. Fast construction of distributed and decomposed evolutionary representation. In *Late Breaking Papers of the Genetic and Evolutionary Computation Conference*, pages 139–148. AAAI Press, 1999.

[23] H. Kargupta and K. Sarkar. Function induction, gene expression, and evolutionary representation construction. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 313–320. AAAI Press, 1999.

[24] Donald Knuth. *The Art of Computer Programming.* Addison-Wesley, Reading, MA, 1968.

[25] S. Kushilevitz and Y. Mansour. Learning decision trees using fourier spectrum. In *Proc. 23rd Annual ACM Symp. on Theory of Computing*, pages 455–464, 1991.

[26] W. Lam and A. M. Segre. Distributed data mining of probabilistic knowledge. In *Proceedings of the 17th International Conference on Distributed Computing Systems*, pages 178–185, Washington, 1997. IEEE Computer Society Press.

[27] W. Lee, S. Stolfo, and Kui Mok. A data mining framework for adaptive intrusion detection. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, page Not available. IEEE Press, 1999.

[28] T. Mitchell. *Machine Learning.* McGraw-Hill, USA, 1997.

[29] Frederick Mosteller and John W. Tukey. *Data Analysis and Regression.* Addison-Wesley, Menlo Park, CA, 1977.

[30] Colm Mulcahy. Plotting and scheming with wavelets. *Mathematics Magazine*, 69(5):323–343, December 1996.

[31] Colm Mulcahy. Image compression using the haar wavelet transform. *Spelman Science and Mathematics Journal*, 1(1):22–31, 1997.

[32] D. Pokrajac, T. Fiez, D. Obradovic, S. Kwek, and Z. Obradovic. Distribution comparison for site-specific regression modeling in agriculture. Published in the 1999 International Joint Conference on Neural Networks, http://www.cas.american.edu/ medsker/ijcnn99/ijcnn99.html, July 1999.

[33] F. J. Provost and B. Buchanan. Inductive policy: The pragmatics of bias selection. *Machine Learning*, 20:35–61, 1995.

[34] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[35] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[36] S. Stolfo et al. Jam: Java agents for meta-learning over distributed databases. In David Heckerman, Heikki Mannila, Daryl Pregibon, and Ramasamy Uthurusamy, editors, *Proceedings Third International Conference on Knowledge Discovery and Data Mining*, pages 74–81, Menlo Park, CA, 1997. AAAI Press.

[37] Eric Stollnitz, Tony D. DeRose, and David H. Salesin. Wavelets for computer graphics: A primer, part 1. *IEEE Computer Graphics and Applications*, 5(3):76–84, May 1995.

[38] Eric Stollnitz, Tony D. DeRose, and David H. Salesin. Wavelets for computer graphics: A primer, part 2. *IEEE Computer Graphics and Applications*, 5(4):75–85, June 1995.

[39] R. et al. Subramonian. An architecture for distributed data mining. In *Advances in Distributed and Parallel Knowledge Discovery*, page Not available. AAAI/MIT Press, 1999.

[40] K.M. Ting and B.T. Low. Model combination in the multiple-data-base scenario. In Maarten van Someren and Garhard Widmer, editors, *Machine Learning: ECML-97*, number 1224 in Lecture Notes in Computer Science : Lecture Notes in Artificial Intelligence, pages 250–265, New York, 1997. Springer-Verlag. 9th European Conference on Machine Learning.

[41] M. V. Wickerhauser. *Adapted Wavelet Analysis from Theory to Software.* A. K. Peters Ltd., 1994.

[42] K. Yamanishi. Distributed cooperative bayesian learning strategies. In *Proceedings of COLT 97*, pages 250–262, New York, 1997. ACM.