A Fourier Analysis Based Approach to Learning Decision Trees in a Distributed Environment

Byung-Hoon Park,
School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164-2752, USA
Email: bhpark@eecs.wsu.edu

Rajeev Ayyagari and Hillol Kargupta
Email: arajeev1@cs.umbc.edu, hillol@cs.umbc.edu
Department of Computer Science and Electrical Engineering,
University of Maryland Baltimore County, 1000 Hilltop Circle,
Baltimore, MD 21250.
Phone: 410-455-3500
Fax: 410-455-3969

Paper Number 207

Abstract

Spurred by advances in communication technologies, mobile computing and databases that are distributed have become widespread. Such a computing environment involves data that is stored at geographically dispersed locations, and the so-called "slim" computing devices such as palmtops and wearable computers. The decentralized nature of data storage and this new paradigm in computing give rise to several issues, such as security, communication overhead, computational load demands and scalability, that are not adequately addressed by traditional centralized data mining techniques.

It is essential that algorithms designed for distributed data mining scenarios mitigate some of these issues. This paper attempts to adapt one centralized data mining technique, decision tree learning, to such an environment. It presents a scalable algorithm that can be used to build decision trees from a distributed, heterogeneous database while minimizing communication overheads. This paper also shows how a decision tree may be represented in terms of its Fourier spectrum. It uses this Fourier spectrum based technique to aggregate decision trees built at the various distributed sites, simplifying the model built during the data mining stage, and notes some additional advantages of the Fourier spectrum approach.

1 Introduction

Information gathering infrastructures have grown steadily in pervasiveness and power in recent years, giving rise to large electronic data storage stations. Large volumes of data are gathered from myriad sources, stored, and utilized for researches into a wide spectrum of disciplines. Reflecting the plethora of fields for which the data is collected, and the distributed nature of the data gathering systems, its storage is highly decentralized in nature at locations that are often widely dispersed.

For example, The NASA Earth Observing System (EOS)[1] generates more than 100 gigabytes of image data per hour, that are stored, managed, and distributed by the EOS data and information system (EOSDIS). The data managed by EOSDIS are available primarily from eight EOSDIS Distributed Active Archive Centers (DAACs). Each of the eight NASA DAACs provides data pertaining to a particular earth science discipline. These DAACs collectively provide a physically distributed but logically integrated database to support interdisciplinary research.

The Internet provides another example of a source of distributed data. A prime source of financial data is financial news stories regularly posted on the Internet, which include announcements regarding new products, quarterly revenue, legal battles and mergers. In addition, sites such as Yahoo finance and CNN finance offer company financial profile data. The websites of various companies and online stock quotes form other sources of such data.

The ability to extract meaningful information from such data is of paramount importance in today's world. Managerial or strategic decision making often depends on the ability to extract such information in a timely manner, without too many restrictions on the type of computing or communication resources immediately available to the decision maker. This paper makes the case that these goals can be achieved by using the paradigm of Distributed Data Mining (DDM). In particular, it develops a distributed decision tree learning algorithm. The algorithm builds a model at various distributed sites, and transmits the model and a modest amount of data to a central site. The data to be transmitted is determined using boosting, a technique that was used to help enhance the appropriateness of the information transmitted to the central site. It is noted that the model built in this fashion was too complicated, and Fourier analysis techniques are developed to help mitigate the problem. The paper shows how the Fourier analysis techniques can be used to simplify the model built, and indicates how to mine data effectively in a way that distributes computational load and reduces communication costs.

Section 2 presents an abstraction of the problem and the motivation behind our approach. It also provides an introduction to decision trees. Section 3 describes the algorithm for building a distributed decision tree model. Section 4 shows how Fourier analysis techniques can be used in conjunction with decision trees and describes a method for simplifying the model built. Section 5 describes experimental verification of the techniques proposed. Finally, section 6 describes avenues for future research and concludes this paper.

2 Background

We use the *relational model* of data storage in this paper. That is, each "piece of data" (also called an *instance, record, row* or *individual*) is a collection of *features*, also called *attributes* or *records*. Attributes may be discrete or continuous. There is a collection of features that, taken together, uniquely identify an individual. This is known as the *key*. Relational data collected and stored in a distributed fashion may be *homogeneous*, where each site stores the same features, but different individuals. However, it is often the case that the data is *heterogeneous*, with the sites storing different features of the *same* individual. The former method of data storage is also called

[1] `http://eospso.gsfc.nasa.gov`

Key	Avg Monthly Visits	Medical History	Hi Risk
1	0.8	bad	Yes
2	0.9	bad	Yes
3	1.0	good	No
4	0.8	good	No
5	0.7	good	No
6	0.6	bad	Yes
7	1.4	good	No
8	1.1	good	Yes
9	1.5	good	No
10	2.1	good	Yes
11	1.0	good	No
12	1.6	good	Yes
13	1.3	good	Yes
14	0.8	good	No
15	1.4	bad	Yes
16	1.0	good	No
17	1.2	good	Yes
18	1.1	good	No
19	1.1	good	No

Key	Avg Med Expenses (Monthly)	Income	Hi Risk
1	53.00	39600	Yes
2	86.75	45500	Yes
3	23.25	60100	No
4	78.50	55000	No
5	55.00	31300	No
6	65.50	37400	Yes
7	28.25	44200	No
8	33.75	28100	Yes
9	14.25	25900	No
10	39.75	38200	Yes
11	27.00	35500	No
12	31.25	41600	Yes
13	29.00	57400	Yes
14	46.00	47100	No
15	27.50	42300	Yes
16	28.25	33000	No
17	33.50	36800	Yes
18	26.25	41300	No
19	25.50	33500	No

Figure 1: Sample data for the insurance problem under (left) medical and (right) financial records.

horizontally partitioned data storage, whereas the term *vertically partitioned* is sometimes used to describe the latter. A common or *key* attribute at the sites that identifies an instance can usually be constructed for heterogeneous data. In this paper, we consider the problem of supervised learning over distributed, heterogeneous data sets. Given a classification of each instance in the data, supposed to be the "correct" classification, we try to determine a function of the features that places each instance into a "class".

Many researchers have developed techniques for analyzing distributed data. Some existing techniques have been adapted to homogeneous data [3, 6, 42]. Several new techniques have also been developed [8, 7, 33, 41, 28, 17, 45, 10, 27, 21, 20, 9, 32, 20, 11]. Of late, algorithms for the analysis of distributed heterogeneous have excited interest. Some of the work in this field includes [34, 1, 44, 31, 18, 19, 24, 23, 22, 25, 43]. A large fraction of the existing research into distributed, heterogeneous data sites concerns unsupervised learning [23, 19, 31, 22]. More recently, the problem of supervised learning on heterogeneous data has been investigated [18, 25, 43]. This paper extends earlier work reported in [25].

We illustrate the problem and the difficulties that arise with an example based on toy data. We consider a medical insurance company which wishes to determine whether a particular client is a high-risk patient before offering insurance against medical expenses. In our simplified scenario, the company bases its predictions on data at two different agencies that store medical records and financial records. Sample data is shown in Figure 1.

The "key" field identifies the patient. A notable aspect is the separation of the features. The data at the two sites contains different features, although the data is about the same patients. The specific prediction model used in this paper is a decision tree, built using the well known algorithm

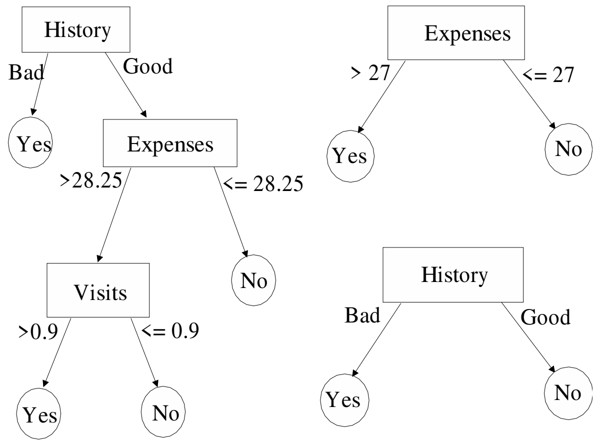

Figure 2: (left) Decision tree computed by C4.5 for the insurance problem when all the data is contained at a single site (top right) Decision tree computed by C4.5 based on partial data from the financial records and (bottom right) from the medical records.

C4.5 [36]. The traditional C4.5 algorithm would produce an accurate decision tree model but would require that the data be centralized, incurring transmission costs, and possibly unacceptable security risks. Using a naive approach, we might build two decision trees, one at each site, by running C4.5 on the data at that site. The three trees obtained using these methods are shown in Figure 2.

While the tree built after centralizing the data has an error of 0%, the other two trees have errors of 26.3%! The reason for this is evident when we look at the centralized decision tree: there is a *cross-term,* an interaction between Average Monthly Visits and Average Monthly Expenses, which cannot be captured at either site. Thus, such naive approaches cannot achieve the objective of accurate prediction. Clearly, there is a trade-off between the accuracy and the amount of data transmitted to a central location. This paper explores a technique that allows the amount of data transmitted and the error rates achieved to be varied in a controlled fashion. It uses ensembles of decision trees constructed using boosting [37, 14, 16] to build a model of the data in a truly distributed fashion, as shown in the next section. It then explores Fourier analysis techniques that can be used to construct a simpler decision tree from this model.

3 Distributed Decision Tree Learning in a Heterogeneous Environment

In this paper, we use decision trees as a model of the data. Decision trees are attractive models, because they are easy to understand and generate simple decision rules. Decision tree learning algorithms such as ID3 [35] and C4.5 [36] are also fast, accurate on many data sets, and easily available. This section shows how to build decision trees in a distributed fashion using boosted ensembles.

Some work has been done on decision tree learning in a distributed, environment. However, the synchronous tree construction approaches considered before are not suitable for the problem we consider. In such an approach, all the sites contribute in selecting a feature at every node of the globally maintained tree. For example, the ID3 algorithm computes the information gain for every choice of feature. So all the sites need to be informed regarding the particular subset of data

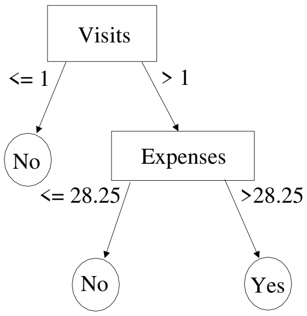

Figure 3: Tree constructed from instances for which both local sites give low confidence

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 $O(ns)$, 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 $O(nsk)$. If the table has c columns and n rows then the cost of moving the complete data sets to a single site is $O(nc)$. When $sk > c$ 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.

To motivate our approach, we return to the example of the medical insurance company. As shown before, decision trees built locally by C4.5 at each of the sites have high prediction error. The decision trees output by C4.5 assign a confidence to each leaf, defined as the proportion of data points falling into that leaf that are correctly classified. If we now look carefully at the particular instances that were misclassified (and hence were responsible for the error) at the sites, we find that they are the ones with the lowest confidence. Indeed, this is how the confidence is defined. Now, we gather the features from both sites for the instances with the lowest confidence at a single central site, and run C4.5 on the data thus obtained. The resulting tree is shown in Figure 3. This tree captures the cross term that was missed by the local trees, but was present in the centralized tree, *viz.* that between Average Monthly Visits and Average Monthly Expenses. In fact, the conditions tested in this tree, built from just a small fraction of the data, are almost identical to those in the centralized tree. This suggests that distributed learning can be accomplished using little data communication.

Now we turn our attention to the question of how these "bad" training instances may be detected. Several techniques were tried. It was found that boosting gave the best selection in terms of the appropriateness of the rows transmitted, as gauged by the cross terms that are evaluated at the central site from this data. Boosting [37, 14, 16] is a technique that improves the performance of any weak learner by modifying the distribution according to which training data is sampled, while leaving the underlying algorithm unaltered. AdaBoost, proposed by Freund and Schapire in [16], is a boosting algorithm that has been studied in a theoretical framework [4, 5, 13, 30, 39] as well as an empirical one [2, 30, 12]. It has been proved that the training error of the AdaBoost algorithm decreases with the number of classifiers in the ensemble. Boosting (and in particular, AdaBoost) has the property that it identifies training instances that are difficult to classify. In this paper, we define a measure for each site, the *confidence* of that site, which is closely tied to the boosting distribution. Thus, a low confidence identifies the "bad" rows. The connection between the confidence measure (defined below) and the boosting distribution is shown in Appendix. The following subsection describes the algorithm in detail.

3.1 The Algorithm

The proposed algorithm offers a way to balance the transmission cost and the error by identifying the training instances which have the most information about cross-terms. This is done using the *confidence* defined below. The technique involves setting a threshold for the confidence. Instances classified with confidence lower than the threshold are transmitted, along with the local models, to a central site.

The specific decision tree algorithm used is the C4.5 algorithm [36]. AdaBoost is used to build an ensemble of decision trees at each local site. The AdaBoost algorithm is used in conjunction with C4.5, as in [15]. At the end of the boosting, each site has an ensemble of decision trees. When presented with an example, each of these trees will return a real number in $[0,1]$. This number represents both the prediction and the confidence of the tree in that prediction. (If it is closer to one, the confidence that the instance is a positive one is higher.) The confidence values contain more information than a binary classification [40].

The algorithm proceeds as follows. First, AdaBoost is used with C4.5 to build an ensemble at each site. Suppose the trees obtained after boosting are $T_{ij}, i = 1, ..., k, j = 1, ..., m_i$, where m_i is the number of boosting iterations at site S_i. Given an example $\mathbf{x}$, tree T_{ij} returns a prediction/classification p_{ij}.

The *confidence* of predicting 0 at site S_i is defined as

$$c_i^0 = \sum_{2p_{ij}-1<0} \alpha_t \left|2p_{ij} - 1\right|,$$

and the confidence of predicting 1 at S_i is similarly defined as

$$c_i^1 = \sum_{2p_{ij}-1>0} \alpha_t (2p_{ij} - 1),$$

where α_t refers to the boosting weight. [38] suggests using

$$\alpha_t = 1/2 \ln\left((1 - \epsilon_t)/\epsilon_t\right)$$

where $\epsilon_t = \Pr_{D_t}(h_t(x_i) \neq y_i)$ is defined to be the error made by the weak learner. This is evaluated as $\epsilon_t = \sum_{i:h_t(i)\neq y_i} D_t(i)$. The prediction of site S_i as a whole is given by $p_i = 0$ if $c_i^0 > c_i^1$ and 1 otherwise.

The confidence of this prediction at site S_i is $c_i = \left|c_i^0 - c_i^1\right|$. As shown in Appendix, this confidence is closely related to the boosting distribution, and hence gives an idea of how easy or hard it is to classify a particular training instance.

The idea used in the algorithm is that the instances that are difficult to classify correctly should be sent to the central site during the model-building stage. Thus, the model at the global site is based on those instances that are difficult to classify. At the classification stage, the instances that are determined as difficult to classify are similarly sent to the global site.

The boosting can be used to determine which instances are difficult to classify. In particular, a row which is classified with very low confidence by every site is difficult, since the boosting would improve the confidence with which that row is classified if it were not.

A training row $\mathbf{x}$ is considered "bad" if $\max_{i=1,...,k} c_i < \gamma_1$, where γ_1 is a threshold that is set externally. (The c_i values are, of course, dependent on the training example $\mathbf{x}$.)

If a row is determined to be bad, features for that row from all the local sites are sent to the global site. At the end of this procedure, the global site has complete information about a fraction of all the training instances. The size of this fraction is determined by the threshold γ_1.

Another boosted model is then built at the global site based on all the transmitted instances, resulting in an ensemble $T_1, T_2, ..., T_m$. This completes the model-building stage of the algorithm.

3.2 Decision Tree Aggregation: Classification of Unseen Instances

This section explores techniques for classifying unseen instances using the local and global models built as above. The exact method used to combine the models is important, as discussed below. Three different techniques were tried. The weighted average and the linear OLS scheme failed to perform satisfactorily. The Tournament scheme performed significantly better than the other two.

The weighted average scheme assigns weights to each local site, computed as $w_i = -\log((1-\epsilon_i)/\epsilon_i)$, where ϵ_i is the training error at site i. We then calculate the *vote* for the classifications 0 and 1 as $w^0 = \sum_{i:p_i=0} w_i$ and $w^1 = \sum_{i:p_i=1} w_i$. If $|w^0 - w^1|$ is above a threshold γ_2, the local sites are used for the prediction. The prediction is 0 if $w^0 - w^1 > 0$ and 1 otherwise. If this *overall confidence* is below the threshold γ_2, the prediction made by the global site model is used.

While the voting or weighted average scheme performs better than the naive approaches (such as using the estimates of the local sites without any kind of aggregation), we found that it failed to perform as well as the Tournament scheme on both datasets presented in the experimental section as well as several others. The Tournament aggregation scheme performs significantly better. This may be because the vote of a local model that is performing consistently well may be "damped" by other poorly performing models in an averaging scheme such as this one. Thus, even if a local partition is the most significant one, the instance may be referred to the global model for classification, leading to incorrect classifications.

The OLS scheme is a variation on weighted averaging. Instead of determining the weights based on the error of the local sites, the predictions of the local sites *and the global site* for the training instances are treated as independent variables, with the correct prediction being the dependent variable. Linear regression is used to determine the weights assigned to each site, as well as a constant.

Assume that $\mathbf{x}_i, i = 1, ..., N$ are the training instances. The model at site S_j predicts the value p_{ij} for training example $\mathbf{x}_i$, and the model at the global site predicts g_i. The model used is $\boldsymbol{Y} = \boldsymbol{X\beta} + \boldsymbol{\epsilon}$,where Y is the $N \times 1$ vector of true classifications, $\boldsymbol{X}$ is the $N \times (k+2)$ matrix $[((p_{ij})) : ((g_i)) : \mathbf{1}]$, $\boldsymbol{\beta}$ is the $(k+1) \times 1$ vector of coefficients (β_{k+2} is the regression constant and the remaining β_is represent the coefficients for each site), and $\boldsymbol{\epsilon}$ is the error vector. $\boldsymbol{\beta}$ is estimated using OLS.

Given an unseen instance $\mathbf{x}$, it is classified as follows. Let p_i be the prediction of site S_i for $\mathbf{x}$ and let g be the prediction of the global site. Then the final prediction is $p = \beta_{k+2} + \sum_{i=1}^{k} \beta_i p_i + \beta_{k+1} g$.

We assume here that there is a linear relationship between the predictions of the local and global sites and the true classifications. This turns out to be an invalid assumption. The OLS scheme failed to perform on both the datasets used in the experiments. The errors for both datasets was close to 50%, i.e. this scheme is not much better than random guessing.

The Tournament scheme attempts to address some of the problems that arise with the other two schemes presented in this section. By using only the best of the local classifiers, the Tournament scheme ensures that any highly significant local partitions get high votes and are not damped out by other partitions. Thus, the chances that unseen instances that should be classified by the local models are referred to the global model are reduced. This scheme also seems to be more amenable to the binary class label situation than the OLS scheme is.

Under the tournament scheme, we use only the *best* local classifier instead of a linear function of all the local classifiers. If the local prediction is found to be unsatisfactory, the global classifier is used.

To classify a new example $\mathbf{x}$, we find the values c_i^0 and $c_i^1, i = 1, ..., k$. Let $c = \max_{i=1,...,k; j=0,1} c_i^j$, and $(I_p, J_p) = \arg\max_{i=1,...,k; j=0,1} c_i^j$. If $c > \gamma_2$, where γ_2 is a preset threshold, then the final prediction is J_p. If not, this row is identified as a "difficult" row to classify, and the ensemble $T_1, ..., T_m$ built at the global site is used to classify this row. This was found to be the best technique in our experiments.

The techniques described in this section show how to build a model comprising a multitude of decision trees. In doing so, we lose one of the primary benefits of using decision trees: simplicity. A large collection of decision trees aggregated using a tournament scheme does not lend itself to easy comprehension. The next section attempts to address this problem by using the Fourier transform of the decision trees to simplify the model built.

4 Fourier Analysis of Decision Trees and its Applications

In this section we describe the tools used to create a single, unified model at the global site from the many models created at the local and global sites. In particular, we describe how compute the Fourier spectrum of a decision tree, and how to build a decision tree efficiently from its Fourier spectrum. We also show how to aggregate the Fourier spectra of several models, that is, how to calculate the Fourier spectrum of an ensemble classifier from the spectra of its individual models. The approach to simplification of the model taken in this section is motivated primarily by the observation, made in [26], that the Fourier spectrum of a decision has an *exponential decay property*, which has the consequence that only a few of the coefficients in the Fourier spectrum need to be computed. We first need to establish notation and set up a mathematical framework under which the aforementioned analysis is carried out.

4.1 The Fourier Spectrum of a Decision Tree

We start with a description of the Fourier basis of the linear space of all real-valued functions on the set of all l-bit boolean feature vectors. Of course, boolean decision trees fall into this category. The Fourier basis is comprised of 2^l Fourier functions, defined as $\psi_{\mathbf{j}}(\mathbf{x}) = (-1)^{\mathbf{x}\cdot\mathbf{j}}$, where $\mathbf{j}, \mathbf{x} \in \{0,1\}^l$. The notation $\mathbf{x}\cdot\mathbf{j}$ represents the inner product of the two vectors $\mathbf{j}$ and $\mathbf{x}$, modulo 2. $\psi_{\mathbf{j}}(\mathbf{x})$ can either be equal to 1 or -1. The string $\mathbf{j}$ is called a *partition*. The *order* of a partition $\mathbf{j}$ is the number of 1-s in $\mathbf{j}$. A Fourier basis function depends on some x_i only when $j_i = 1$. Therefore a partition can also be viewed as a representation of a certain subset of x_i-s; every unique partition corresponds to a unique subset of x_i-s. If a partition $\mathbf{j}$ has exactly α number of 1-s then we say the partition is of order α since the corresponding Fourier function depends on only those α number of variables corresponding to the 1-s in the partition $\mathbf{j}$. A function $f : \mathbf{X}^l \rightarrow \Re$, that maps an l-dimensional space of binary strings to a real-valued range, can be written as

$$f(\mathbf{x}) = \sum_{\mathbf{j}} w_{\mathbf{j}} \psi_{\mathbf{j}}(\mathbf{x}),$$

where $w_{\mathbf{j}}$ is the Fourier Coefficient corresponding to the partition $\mathbf{j}$;

$$w_{\mathbf{j}} = \frac{1}{2^l} \sum_{\mathbf{x}} f(\mathbf{x}) \psi_{\mathbf{j}}(\mathbf{x}).$$

We note that $\sum_{\mathbf{x}\in\mathbf{X}} \psi_{\mathbf{j}}(\mathbf{x}) = 0$. The Fourier coefficient $w_{\mathbf{j}}$ can be viewed as the relative contribution of the partition $\mathbf{j}$ to the function value of $f(\mathbf{x})$. Therefore, the absolute value of $w_{\mathbf{j}}$ can be used as the "significance" of the corresponding partition $\mathbf{j}$. If the magnitude of some $w_{\mathbf{j}}$ is very small compared to other coefficients then we may consider the $\mathbf{j}$-th partition to be insignificant and neglect its contribution.

We are now ready to describe the exponential decay property and a method for calculating the Fourier coefficients efficiently from a decision tree. In this paper, we indicate how the exponential decay property may be proved for decision trees with non-boolean features. The techniques for calculating the Fourier spectrum are described for boolean decision trees, but can be modified easily to suit the non-boolean case. Following the notation introduced above, the Fourier transform can be represented as

$$w_{\mathbf{j}} = \frac{1}{|\Lambda|}\sum_{\mathbf{x}\in\Lambda} f(\mathbf{x})\psi_{\mathbf{j}}(\mathbf{x}) = \frac{1}{|\Lambda|}\sum_{i=1}^{n}\sum_{\mathbf{x}\in S_{l_1}} f(\mathbf{x})\psi_{\mathbf{j}}(\mathbf{x}) = \frac{1}{\Lambda}\sum_{i=1}^{n}|S_{l_i}|f(\mathbf{h_i})\psi_{\mathbf{j}}(\mathbf{h_i}),$$

where Λ denotes the instance space, S_{l_i} is the subset of Λ whose points fall into the i^{th} leaf node and $\mathbf{h_i}$ is the *schema defined by a path to* l_i. This simply means that we have a schema in which the features that are not fixed on the path to the leaf nodes have $*$-s in their place.

The following properties were first stated, in a different form, in [29]. Firstly, if a feature in a partition $\mathbf{j}$ is non-zero, and the same feature has a $*$ or don't-care value in the schema of a leaf l_i, then $\sum_{\mathbf{x}\in S_{l_i}} \psi_{\mathbf{j}}(\mathbf{x}) = 0$. Among the implications of this is that Fourier coefficients corresponding to schemata with order greater than the depth of the tree are zero. This result eliminates a great deal of computation. Secondly, the *energy*, or the sum of squared coefficients, of the coefficients decreases exponentially with their order. A stronger result is proved in [29], *viz.* $\sum_{|\mathbf{j}|\geq k} w_{\mathbf{j}}^2 < g(k)$ where $g(k)$ is decreases exponentially in k. That paper also gives a closed form for $g(k)$. To indicate how the results of this section adapt to the non-binary case, we extend this result to the non-binary case below. As mentioned above, the other results will be stated only for the binary case.

To prove this result for the non-boolean case, consider the function $\kappa : [\lambda]^l \to \mathcal{R}$, where the notation $[\lambda]$ stands for the set $\{0, 1, ..., \lambda - 1\}$. For the purposes of this section, assume further that $\lambda = 2^q$, so that each non-binary variable can be represented using q bits. Let $b : [\lambda]^l \to [2]^{ql}$ be the canonical map from the non-boolean feature space to the boolean feature space. The Fourier basis functions for the non-boolean case are

$$\phi_{\mathbf{j}}(\mathbf{x}) = \prod_{m=1}^{l} \exp^{\frac{2\pi i}{\lambda} x_m j_m}$$

We have

$$\sum_{\mathbf{j}\in[\lambda]^l} \eta_{\mathbf{j}}\phi_{\mathbf{j}}(\mathbf{x}) = \sum_{\mathbf{j}\in[2]^{ql}} w_{\mathbf{j}}\psi_{\mathbf{j}}(b(\mathbf{x})).$$

Here $\eta_{\mathbf{j}}$ are the non-boolean Fourier coefficients. Taking the inner product of both sides of this equation with $\phi_{\mathbf{i}}$ yields

$$\eta_{\mathbf{i}} = \sum_{\mathbf{j}\in[2]^{lq}} w_{\mathbf{j}} \sum_{\mathbf{x}\in[\lambda]^l} \psi_{\mathbf{j}}(b(\mathbf{x}))\phi_{\mathbf{i}}(\mathbf{x}).$$

Now, the expression $\sum_{\mathbf{x}\in[\lambda]^l} \psi_{\mathbf{j}}(b(\mathbf{x}))\phi_{\mathbf{i}}(\mathbf{x})$ equals zero whenever $|\mathbf{j}| < |\mathbf{i}|$, where $|\cdot|$ denotes the order of the partition, because a nonzero partition corresponding to a non-boolean feature must map to a non-zero partition in the boolean case. (Analytically, this is because the sum will add over all λ roots of unity for each value of $\psi_{\mathbf{j}}(b(\mathbf{x}))$.)

Thus, we have

$$\sum_{|\mathbf{i}|\geq k} |\eta_{\mathbf{i}}|^2 = \left|\sum_{|\mathbf{j}|\geq k} w_{\mathbf{j}} \sum_{\mathbf{x}\in[\lambda]^l} \psi_{\mathbf{j}}(\kappa(\mathbf{x}))\phi_{\mathbf{i}}(\mathbf{x})\right|^2 \leq \sum_{|\mathbf{j}|\geq k} w_{\mathbf{j}}^2 \left|\sum_{\mathbf{x}\in[\lambda]^l} \psi_{\mathbf{j}}(\kappa(\mathbf{x}))\phi_{\mathbf{i}}(\mathbf{x})\right|^2 \leq K \sum_{|\mathbf{j}|\geq k} w_{\mathbf{j}}^2 \leq K g(k)$$

where K is a constant that bounds the inner sum. (g is the bounding function defined in [29].) Thus, the non-boolean Fourier spectrum also has the exponential decay property.

```
Function Detect(input:  Decision Tree T)
  for each possible enumeration path i to leaf nodes of T
    w_i ← Calculate(w_i)
  end
end
```

Figure 4: *Algorithm for obtaining Fourier spectrum of a decision tree.*

4.2 Decision Tree Aggregation Using the Fourier Spectrum

The results mentioned thus far show that the Fourier spectrum can be computed efficiently. The exponential decay property tells us that the magnitude of the high-order coefficients drops off rapidly, so that it is enough to calculate a few lower order coefficients. These results also provide us with a technique for calculating the Fourier coefficients. The results in this section explain how we can use the Fourier spectrum to simplify the decision tree model by reducing the number of decision trees.

We first note that the final classification output by each local site, the boosted classification, is a weighted average of the individual trees at that site. Thus, the Fourier spectrum of the combined classifier can be obtained simply as a weighted linear combination of the spectra of the individual trees. The algorithm for calculating the Fourier spectrum of a tree is shown in Figures 4 and 5. The algorithm shown calculates *all* non-zero coefficients. Alternatively, a cut-off for the order of the coefficients could be selected, based on the energy, to reduce the number of coefficients evaluated further. We can calculate the Fourier spectrum of the combined classifier as

$$f(\mathbf{x}) = \sum_{i=1}^{n} a_i f_i(\mathbf{x}) = \sum_{i=1}^{n} a_i \sum_{\mathbf{j} \in J_i} w_{\mathbf{j}} \psi_{\mathbf{j}}(\mathbf{x}),$$

where $f_i(\mathbf{x})$ and a_i are the i-th decision tree and its weight and J_i is set of all its non-zero coefficients.

In order to build a single tree that is equivalent to the local classifier, we need an efficient technique to build a decision tree from its Fourier spectrum. The algorithm we propose does this by simulating the working of the ID3 algorithm, which selects the attribute a_i with maximum information gain. In other words, to select an attribute a_i for the next node, we need to measure entropy of each subtree branched by a_i. Since each such subtree can be denoted as a schema, evaluation of entropy reduction by choosing a_i can be measured by calculating the average of all those schemata with a_i at the node. This schema average is essentially a measure of the distribution of class labels. Therefore, the entropy is considered to be low if the average is close to either one or zero. A more detailed description of the efficient calculation from the Fourier coefficients is provided in Kargupta et al [25]. Figures 6 and 7 give the algorithm for constructing a decision tree from its Fourier spectra.

This simplifies the local model at each site. The advantage is that we have a more comprehensible model than the one we had before. However, we still do not have a single decision tree that is equivalent to the entire model. To do this, we need to be able to efficiently aggregate Fourier spectra *across the heterogenous sites and the global site*. The following sketches a global decision tree building algorithm which is currently under development.

Under the tournament-based distributed classifier system, a classification of an instance x is chosen based on the highest confidence $f(x)$ among all classifier sites. Following the same rationale, the algorithm estimates the entropy of an schema $\mathbf{h}$ by choosing the minimum among those returned by all classifier sites. If the entropy of $\mathbf{h}$ is minimum at classifier site C_i, it indicates that we can predict the classification of $\mathbf{h}$ with probability of schema average at site C_i. It also ensures that every data instance $x \in \mathbf{h}$ is likely to be assigned the classification from C_i with high probability since a schema average is the expected confidence.

Function Calculate(input: Fourier Coefficient w_i)
  $w_i \leftarrow 0$;
  for each leaf node l_j
    $\mathbf{h_l} \leftarrow$ a schema defined by a path to l_j
    $w_i \leftarrow w_i + |S_{l_j}| \times label(l_j) \times \psi_i(\mathbf{h_l})$
  end
  $w_i \leftarrow w_i / |\Lambda|$
end

Figure 5: *Algorithm for calculating a Fourier coefficient.* Λ *denotes the instance space,* S_{l_j} *is the subset of* Λ *whose points fall into the* j^{th} *leaf node*

Function BestAttribute(input: Schema $\mathbf{h}$)
  for each attribute $attr_l$
    for each possible value of $attr_l$
      $\mathbf{h_l} \leftarrow$ update $\mathbf{h}$ with $attr_l$
      $e_l \leftarrow 0$
      for each Fourier spectrum FT_i
        $e_l \leftarrow e_l + a_i \times average(FT_i, h_l)$
      end
    end
    $e_l \leftarrow min(e_l, 1 - e_l)$
  end
  $best\_attr \leftarrow$ attribute $attr_l$ with minimum e_l
  return $(best\_attr, e_l)$
end

Figure 6: *Algorithm for Choosing the Best Attribute.* a_i *is the weight of* i^{th} *decision tree and* $average(FT_i, \mathbf{h_l})$ *returns the average of schema* h_l *using* i^{th} *Fourier spectrum.*

So far, this section has presented the Fourier spectrum approaches as tools for simplifying the decision tree model built. We would like to point out that this is by no means the only use for the Fourier spectrum in distributed learning. The exponential decay property of the Fourier spectrum ensures that transmitting the Fourier coefficients directly from the local sites would not be too costly an operation. Fourier coefficients can be estimated directly from the data [25]. This is particularly true if the data is uniformly distributed over the domain of the function we are trying to learn. An accurate Fourier representation can be built up easily in this case. Since the Fourier representation is concise, it proves cost-effective to transmit the coefficients directly to the central site. Once this is done, the Fourier coefficients of cross-terms can be estimated by transmitting a pre-determined number of data instances to the central site and estimating the cross-terms based on these (using a technique like linear regression).

5 Experiments

This section describes experimental verification of the techniques discussed in this paper. The exponential decay property and the validity of the Fourier techniques described in Section 3 are demon-

```
Function Construct(input: Schema h,Threshold δ)
  Create a root node
  (best_attr, entr) ← BestAttribute(h)
  root ← best_attr
  if ( entr < δ )
    return root
  end
  for each possible branch br_i of root
    h_i ← update h with br_i
    Construct(h_i, δ)
  end
end
```

Figure 7: *Algorithm for Constructing a Decision Tree Using Fourier Spectrum.*

strated. The performance of the model-building and aggregation techniques described in Sections 4 and 5 are investigated.

5.1 Datasets

The experiments were performed on two datasets: Quest and Promoter Gene Sequence. Both datasets had binary class labels.

The Gene Sequence data was artificially generated using a decision tree built by C4.5 when it was executed on the original Promoter Gene Sequence data set obtained from the UCI Machine Learning Repository[2]. The data used to build the tree had 76 instances with 57 attributes. Each attribute had 4 four possible values: a,c,g,t. The decision tree thus obtained was then queried repeatedly for the classification of 55000 randomly generated gene sequences, and random noise was added to the classifications. Of the 55000 instances generated, 50000 were used for training and 5000 for testing.

The Quest Data Generator[3] was used to synthetically generate 20000 training and 2000 test instances with 40 attributes. The attributes were all taken to be continuous and no discretization was performed.

5.2 Results

The results of the experiments are shown in the graphs in Figure 8. The first two graphs are for experiments performed on the Quest data. The next two graphs are for 0 on the Promoter Gene Sequence data. Both datasets were split into 5 sites, selecting the features at random for each site. This represents a high degree of fragmentation. The fifth graph demonstrates the exponential decay property of the Fourier coefficients of a decision tree. The last graph shows the behaviour of a the Fourier representation of the entire model when coefficients of a certain order are used.

For the Quest and Gene Sequence data, the model-generating algorithm and the Tournament aggregation technique were run for various values of the thresholds γ_1 and γ_2. Graphs a. and c. show the accuracy of the algorithms for various values of the transmission rate, for the two data sets. Both show an almost linear trend.

Graphs b. and d. give an indication of how well the boosting-based technique identifies the cross terms (i.e. the badly classified rows). Note that the local sites perform very well on the instances

[2] `http://www.ics.uci.edu/~mlearn/MLRepository.html`
[3] `http://www.almaden.ibm.com/cs/quest/syndata.html`

that they classify, but comparatively poorly on the set of all training instances. These graphs are *not* intended to indicate any convergence of the performance of the local sites as the fraction of data they classify increases. They emphasize the difference in performance of the local sites when they are used to classify all the rows and just the "good" rows, or the ones that are not determined as bad. This indicates that those training instances that are being sent to the global site for classification are the ones that *should* be classified by the global model. The appropriate instances are being detected as bad instances.

Graph e. shows the energy of the Fourier coefficients for the Gene sequence problem. The decision trees for this graph were obtained by splitting the data into two sites and running decision tree algorithm on it. The Fourier spectrum of the decision trees so constructed was calculated using techniques described in Section 3. The graph shows a clear exponential decay.

Graph f. shows the accuracy when predicting using the spectrum computed from these decision trees directly. The estimation was done for various orders of coefficients and the accuracy for each order of coefficients used is shown. The graph shows that after a stage, which is reached for even small order coefficients, there is not much improvement in the accuracy. This is a consequence of the exponential decay property of the coefficients. It indicates that accurate learning can be done using only a few lower order coefficients.

The experimental results that the Fourier analysis techniques proposed in Section 3 do indeed perform well on actual data. The boosting-based method for detecting the partitions which need to be estimated at the global site is also shown to be working well. Thus, these techniques form a promising suite of methods for analysis of distributed data in a consistent fashion.

6 Conclusions and Future Work

A scalable, robust approach for the analysis of distributed heterogeneous data has become essential in the field of DDM. The paper presents a method for learning ensembles of decision trees in such an environment. It notes that boosting provides one method of determining which instances contain most cross-term information. The observation that the model built was too complicated to be understood easily drove the next section, which showed how to calculate the Fourier spectrum of a decision tree and indicated how it could be used to simplify the model built. The experimental results showed that the techniques proposed are indeed effective in reducing data communication with acceptable increases to the error.

The Fourier analysis techniques of this paper have a wide variety of potential applications. Some of these applications involve improvements to other algorithms, as demonstrated in this paper. The Fourier coefficients can also be used to test the correlations among certain attributes of data without constructing a whole tree. This may find applications in many areas in a mobile computing environment, where building the entire decision tree may not be feasible. These and other applications of the Fourier techniques outlined in this paper need to be explored in greater detail. In addition to the applications of the Fourier analysis techniques, work needs to be done on improving techniques for aggregation of the decision trees.

Acknowledgments

This research is supported by the United States National Science Foundation grant IIS-0093353.

Appendix: Confidence and the Boosting Distribution

This appendix establishes a connection between the *confidence* used in this paper and the boosting distribution. Following [40], we first make the following definitions. Let $T_0 = \{(x_1, y_1), ..., (x_m, y_m)\}$ be a set of training examples. The weak learner accepts a set of training examples T_0 and returns a hypothesis h. In general, h is a function $h : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ is the instance space and $\mathcal{Y}$ is the classification space. For convenience purposes, we consider the function h defined by $h(x_i) = 2p_i - 1$, where p_i is the prediction of the i-th weak learner (tree, in our case). Thus, h returns either -1 or 1 and our classification space $\mathcal{Y}$ is transformed to $\{-1, 1\}$. In AdaBoost, an initial uniform distribution D over the training examples is assumed. The distribution is updated using the update rule:

$$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

where Z_t is a normalization factor, chosen so that D_{t+1} will be a distribution, and α_t is a weight. The final hypothesis is

$$H(x) = \operatorname{sgn}\left(\sum_{t=1}^{K} \alpha_t h_t(x)\right)$$

As mentioned in [15], the boosting distribution D_t gives higher weight to those instances which are hard to classify and lower weight to those instances which are hard to classify. It is easy to see that

$$D_{T+1}(i) = \frac{\exp(-y_i \sum_{t=1}^{T} \alpha_t h_t(x_i))}{m \prod_{t=1}^{T} Z_t}.$$

Now we define the confidence of the ensemble of classifiers to be $p = \left|\sum_{t=1}^{T} \alpha_t h_t(x_i)\right|$. Thus, we have

$$D_{T+1}(x_i) = \begin{cases} \exp(-p)/K & \text{if classification } = y_i \\ \exp(p)/K & \text{otherwise} \end{cases}$$

Here $K = m \prod_{t=1}^{T} Z_t$ does not depend on the training instance and can be taken as a constant for the current discussion. Thus, for instances that are *correctly classified* by the ensemble of trees, the boosting distribution is an exponentially decreasing function of the confidence. However, as proved in [40], the training error is low if AdaBoost is run for sufficiently many iterations. Thus, the boosting distribution and confidence are related as above for *most* of the training examples.

References

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

[2] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1–2):105–139, 1999.

[3] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[4] L. Breiman. Bias, variance and arcing classifiers. Technical Report 460, Statistics Department, University of California at Berkeley, 1996.

[5] L. Breiman. Prediction games and arcing classifiers. Technical Report 504, Statistics Department, University of California at Berkeley, 1997.

[6] L. Breiman. Pasting small votes for classification in large databases and on-line. *Machine Learning*, 36(1–2):85–103, 1999.

[7] P. K. 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, November 1993.

[8] P. K. 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, Princeton, NJ, 1993.

[9] D. W. 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, December 1996.

[10] V. Cho and B Ẃuthrich. 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 Artifical Intelligence, pages 376–377, New York, 1998. Springer-Verlag. Second Pacific-Asia Conference, PAKKD-98, Melbourne, Australia, April 1998.

[11] V. Crestana and N. Soparkar. Mining decentralized data repositoies. Technical Report CSE-TR-385-99, Ann Arbor, MI, 1999.

[12] T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. *Machine Learning*, 40(2):139–158, 2000.

[13] P. Domingos. Why does bagging work? a bayesian account and its implications. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 155–158, Newport Beach, CA, 1997. AAAI Press.

[14] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

[15] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148–156, Murray Hill, NJ, 1996.

[16] Y. Freund and R. E. Schapire. A decision theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):97–119, August 1997.

[17] Y. Guo and J. Sutiwaraphun. Distributed learning with knowledge probing: A new framework for distributed data mining. In *Advances in Distributed and Parallel Knowledge Discovery.* MIT Press, 2000.

[18] 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. To be published in the Special Issue on Parallel and Distributed Data Mining of the Journal of Parallel Distributed Computing, Guest Eds: Vipin Kumar, Sanjay Ranka, and Vineet Singh.

[19] E. Johnson and H. Kargupta. Collective, hierarchical clustering from distributed, heterogeneous data. In *Lecture Notes in Computer Science*, volume 1759. Springer-Verlag, 1999.

[20] H. Kargupta, I. Hamzaoglu, and B. Stafford. Scalable, distributed data mining - an agent architecture. In D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, editors, *Proceedings Third International Conference on Knowledge Discovery and Data Mining*, pages 211–214, Menlo Park, CA, 1997. AAAI Press.

[21] 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.

[22] H. Kargupta, W. Huang, S. Krishnamrthy, H. Park, and S. Wang. Collective principal component analysis from distributed, heterogeneous data. In D. Zighed, J. Komorowski, and J. Zytkow, editors, *Proceedings of the Principles of Data Mining and Knowledge Discovery Conference*, volume 1910, pages 452–457, Berlin, September 2000. Springer. Lecture Notes in Computer Science.

[23] H. Kargupta, W. Huang, S. Krishnamurthy, and E. Johnson. Distributed clustering using collective principle compent analysis. In *Workshop on Distributed and Parallel Knowledge Discovery*, Boston, MA, USA, 2000.

[24] 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.

[25] H. Kargupta, B. Park, D. Hershberger, and E. Johnson. Collective data mining: A new perspective toward distributed data mining. In H. Kargupta and P. Chan, editors, *Advances in Distributed and Parallel Knowledge Discovery*, pages 133–184. AAAI/ MIT Press, Menlo Park, California, USA, 2000.

[26] 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.

[27] 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.

[28] W. Lee, S. Stolfo, and K. Mok. A data mining framework for adaptive intrusion detection. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, 1999.

[29] N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, fourier transform, and learnability. *Journal of the ACM*, 40:607–620, 1993.

[30] R. Maclin and D. Opitz. An empirical evaluation of bagging and boosting. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence*, pages 546–551, Cambridge, MA, 1997. AAAI Press / MIT Press.

[31] S. McClean, B. Scotney, and Kieran Greer. Clustering heterogeneous distributed databases. In *Workshop on Distributed and Parallel Knowledge Discovery*, Boston, MA, USA, 2000.

[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/ijcc99.html, July 1999.

[33] A. L. Prodromidis, S. J. Stolfo, and P. K. Chan. Effective and efficient pruning of meta-classifiers in a distributed data mining system. Technical Report CUCS-017-99, Department of Computer Science, Columbia University, 1999.

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

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

[36] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kauffman, 1993.

[37] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[38] R. E. Schapire. Theoretical views of boosting. In *Computational Learning Theory: Fourth European Conference, EuroCOLT'99*, pages 1–10, 1999.

[39] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In *Machine Learning: Proceedings of the Fourteenth International Conference*, 1997.

[40] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, pages 297–336, 1999.

[41] S. J. Stolfo, A. L. Prodromidis, S. Tselepis, W. Lee, D. W. Fan, and P. K. Chan. JAM: Java agent to meta-learning over distributed databases. In David Heckerman, Heikki Mannila, Daryl Pregibon, and Ramasamy Uthurusamy, editors, *Proceedings on the Third International Conference on Knowledge Discovery and Data Mining*, pages pages 748–1, Newport Beach, CA, August 1997. AAAI Press.

[42] K. M. Ting and B. T. Low. Model combination in the multiple-data-base scenario. In *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.

[43] K. Tumer and J. Ghosh. Robust order statistics based ensemble for distributed data mining. In *Advances in Distributed and Parallel Knowledge Discovery*, pages 185–210. AAAI/ MIT Press, Menlo Park, California, USA, 2000.

[44] A. L. Turinsky and R. L. Grossman. A framework for finding distributed data mining strategies that are intermediate between centralized strategies and in-place strategies. In *Workshop on Distributed and Parallel Knowledge Discovery*, Boston, MA, USA, 2000.

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

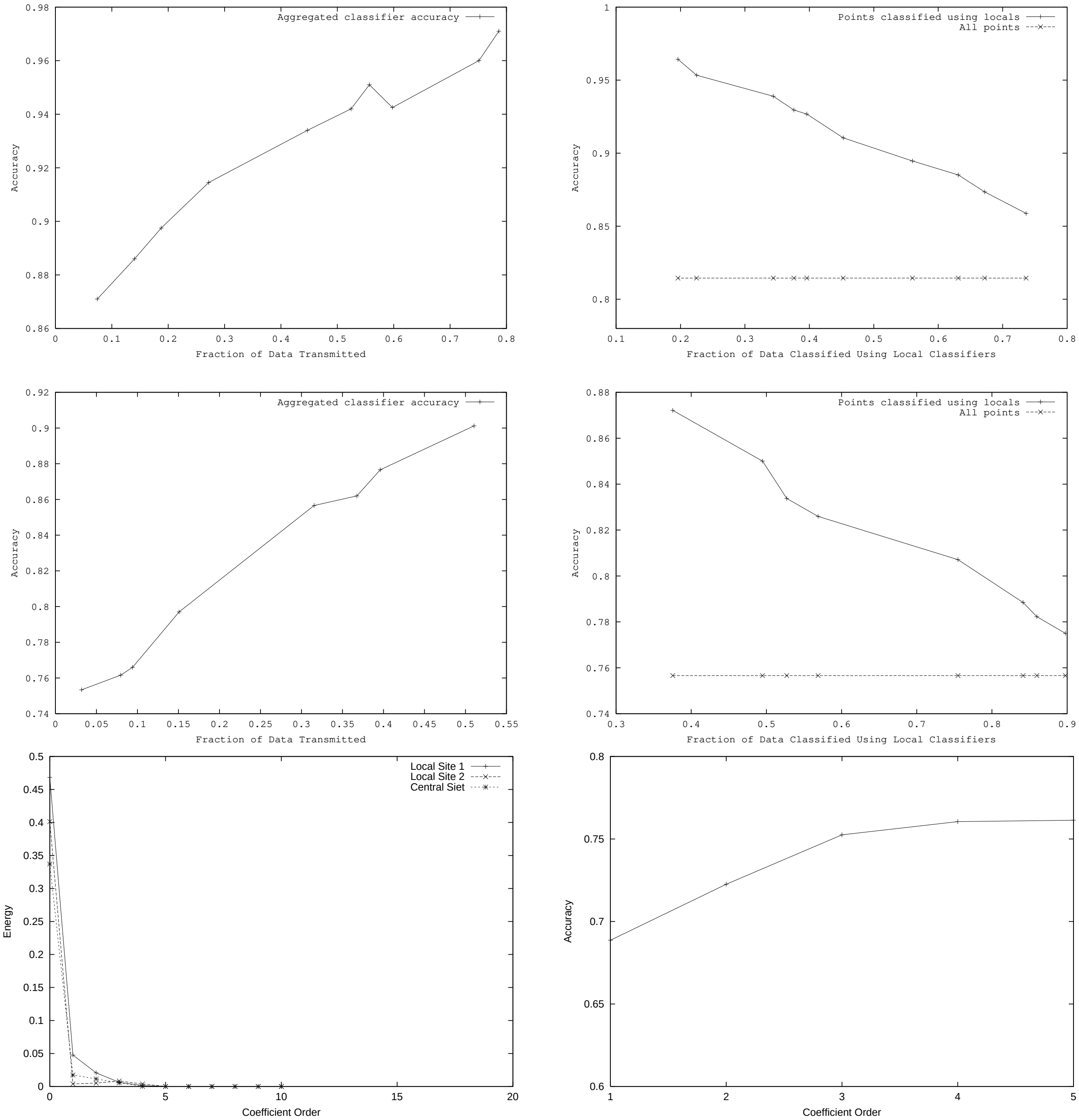

Figure 8: The Figures (left to right and top to bottom) are: a. Accuracy vs. transmission for Quest data, b. Accuracy of local models vs. 0 classified by them, c. Accuracy vs. transmission for Gene data, d. Accuracy of local models vs. instances classified by them, e. Energy of coefficients vs. their order, f. Accuracy of Fourier Representation based model vs. order of coefficients used.