A Fourier Spectrum-based Approach to Represent Decision Trees for Mining Data Streams in Mobile Environments*

Hillol Kargupta and Byung-Hoon Park
Department of Computer Science and Electrical Engineering
1000 Hillltop Circle, University of Maryland Baltimore County
Baltimore, MD 21250, USA,
hillol@cs.umbc.edu, bpark1@cs.umbc.edu

Abstract

This paper presents a novel Fourier analysis-based technique to aggregate, transmit, and visualize decision trees in a mobile environment. Fourier representation of a decision tree has several interesting properties that are particularly useful for mining continuous data streams from small mobile computing devices. This paper presents algorithms to compute the Fourier spectrum of a decision tree and the vice versa. It offers a framework to aggregate decision trees in their Fourier representations. It also describes MobiMine, a mobile data mining system for mining stock-market data from handheld devices connected over low-bandwidth wireless networks.

Key Words: Mobile data mining, decision trees, Fourier spectrum.

1 Introduction

Analyzing and monitoring time-critical data streams using mobile devices in a ubiquitous manner is important for many applications in finance, defense, process control and other domains. These applications demand the ability to quickly analyze large amount of data. Decision trees (e.g., CART[7], ID3[34], and C4.5 [35]) are fast, and scalable. So decision tree-based data mining is a natural candidate for monitoring data streams from ubiquitous devices like PDAs, palmtops, and wearable computers. However, there are several problems.

Mining time-critical data streams usually requires on-line learning that often produces a series of models [11, 13, 25, 36] like decision trees. From a data mining perspective it is important that these models are compared with each other and aggregated if necessary. This is because different data blocks observed at different time frames may generate different models that may actually belong to a single simpler model which can be generated when all the data blocks are combined and mined together. Even for decision trees [10, 40] that are capable of incrementally modifying themselves based on new data, in many applications (e.g., multiple data streams observed at different distributed locations)[29] we end up with an ensemble of trees. Apart from a better understanding of the model, communication of large number of trees over a wireless network also poses a major problem. We need on-line data mining algorithms that can easily aggregate and evolve models in an efficient representation.

*Patent application pending.

Visualization of decision trees [1] in a small display is also a challenging task. Presenting a decision tree with even a moderate number of features in a small display screen is not easy. Since the number of nodes in a decision tree may grow exponentially with respect to the number of features defining the domain, drawing even a small tree in the display area of a palmtop device or a cell phone is a difficult thing to do. Reading an email in a cell phone is sometimes annoying; so imagine browsing over a large number of tree-diagrams in a small screen. It simply does not work. We need an alternate approach. We need to represent trees in such a way that they can be easily and intuitively presented to the user using a small mobile device.

This paper takes a small step toward that possibility. It considers manipulation and visualization of decision trees for mining data streams from small computing devices. It points out that the Fourier bases offer an interesting representation of decision trees that can facilitate quick aggregation of a large number of decision trees [19, 29] and their visualization in a small screen. The efficient representation of decision trees in Fourier representation also allows quicker transmission of tree ensembles over low-bandwidth wireless networks. Although we present the material in the context of mobile devices, the approach is also useful for desktop applications.

Section 2 explains the relation between Fourier representation and decision trees. It also presents algorithms to compute the Fourier spectrum of a decision tree and the vice versa. Section 3 considers aggregation of multiple trees in Fourier representation. Section 4 describes the MobiMine, a mobile data mining system for monitoring the stock market data; it describes the modules that make use of the Fourier representation for aggregation, transmission, and visualization of decision trees. Section 5 documents the performance of the proposed technique in a controlled environment using financial data streams. Finally, Section 6 concludes this paper.

2 Decision Trees as Numeric Functions

This paper adopts an algebraic perspective of decision trees. Note that a decision tree is a function that maps the domain members to a range of class labels. Sometimes, it is a symbolic function where features take symbolic (non-numeric) values. However, a symbolic function can be easily converted to a numeric function by simply replacing the symbols with numeric values in a consistent manner. See Figure 1 for an example. A numeric function-representation of a decision tree may be quite useful. For example, we may be able to aggregate a collection of trees (often produced by ensemble learning techniques) by simply performing basic arithmetic operations (e.g. adding two decision trees, weighted average) in their numeric representations. Later in this paper we will see that a numeric representation is also suitable for visualization and efficient transmission of decision trees in a mobile environment.

Once the tree is converted to a numeric discrete function, we can also apply any appropriate analytical transformation that we want. Fourier transformation is one such possibility and it is an interesting one. Fourier bases offer an additively decomposable representation of a function. In other words, the Fourier representation of a function is a weighted linear combination of the Fourier basis functions. The weights are called Fourier coefficients. The coefficients completely define the representation. Each coefficient is associated with a Fourier basis function that depends on a certain subset of features defining the domain of the data set to be mined. The following section presents a brief review of Fourier representation.

2.1 A Brief Review of the Fourier Basis

Fourier bases are orthogonal functions that can be used to represent any discrete function. Consider the set of all ℓ-dimensional feature vectors where the i-th feature can take λ_i different categorical

values, $\{0, 1, \cdots \lambda_i - 1\}$. The Fourier basis set that spans this space is comprised of $\Pi_{i=0}^{\ell} \lambda_i$ basis functions. Each Fourier basis function is defined as $\psi_{\mathbf{j}}^{\overline{\lambda}}(\mathbf{x}) = \Pi_{m=1}^{l} \exp^{\frac{2\pi i}{\lambda_m} x_m j_m}$, where $\mathbf{j}$ and $\mathbf{x}$ are strings of length ℓ; x_m and j_m are m-th attribute-value in $\mathbf{x}$ and $\mathbf{j}$, respectively; $x_m, j_m \in \{0, 1, \cdots \lambda_m - 1\}$ and $\overline{\lambda}$ represents the feature-cardinality vector, $\lambda_0, \cdots \lambda_\ell$. We often call $\psi_{\mathbf{j}}^{\overline{\lambda}}(\mathbf{x})$ as the $\mathbf{j}$-th basis function. The string $\mathbf{j}$ is called a *partition*, and the *order* of a partition $\mathbf{j}$ is the number of non-zero feature values in $\mathbf{j}$. A Fourier basis function depends on some x_i only when $j_i \neq 0$. If a partition $\mathbf{j}$ has exactly α number of non-zeros values, then we say the partition is of order α since the corresponding Fourier function depends only on those α number of variables that take non-zero values in the partition $\mathbf{j}$.

A function $f : \mathbf{X}^{\ell} \rightarrow \Re$, that maps an ℓ-dimensional discrete domain to a real-valued range, can be represented using the Fourier basis functions: $f(\mathbf{x}) = \sum_{\mathbf{j}} w_{\mathbf{j}} \overline{\psi}_{\mathbf{j}}^{\overline{\lambda}}(\mathbf{x})$, where $w_{\mathbf{j}}$ is the Fourier Coefficient (FC) corresponding to the partition $\mathbf{j}$ and $\overline{\psi}_{\mathbf{j}}^{\overline{\lambda}}(\mathbf{x})$ is the complex conjugate of $\psi_{\mathbf{j}}^{\overline{\lambda}}(\mathbf{x})$; $w_{\mathbf{j}} = \Pi_{i=1}^{l} \frac{1}{\lambda_i} \sum_{\mathbf{x}} \psi_{\mathbf{j}}^{\overline{\lambda}}(\mathbf{x}) f(\mathbf{x})$. 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, we may consider the $\mathbf{j}$-th partition to be insignificant and neglect its contribution. The *order* of a Fourier coefficient is nothing but the order of the corresponding partition. We shall often use terms like *high order* or *low order* coefficients to refer to a set of Fourier coefficients whose orders are relatively large or small respectively.

The Fourier representation of functions defined over Boolean domain is sometimes easier to follow because of its relatively simpler form. Therefore, let us specialize the Fourier bases for Boolean feature space.

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

where $(\mathbf{j} \cdot \mathbf{x})$ is the inner product of $\mathbf{j}$ and $\mathbf{x}$. A partition in the Boolean domain can be viewed as a representation of a subset of features (x_i-s). Every unique partition corresponds to a unique subset of features. In an ℓ-dimensional Boolean feature space there are 2^{ℓ} different subsets of features and therefore the Fourier representation also has 2^{ℓ} different Fourier coefficients. Although the techniques presented in this paper are not restricted to Boolean domain, we would use the above expressions through out this paper for illustrating the ideas.

2.2 Computing the Fourier Transform of a Tree

A decision tree computes the class label of a domain member by traversing through the paths in the tree from the root to a leaf node. Although the Fourier representation can handle real-valued range, in this paper we shall consider functions with discrete range defined by the class labels of the classification problem. In order to keep the formulation simple we shall consider only Boolean classification problems. So the decision tree can be treated as a function f that maps the discrete ℓ-dimensional domain to $\{0, 1\}$.

Now recall that any node in a decision tree is associated with some feature x_i. A downward link from the node x_i is labeled with an attribute value of this i-th feature. As a result, a path from the root node to a successor node represents the subset of domain that satisfies the different feature values labeled along the path. These subsets are essentially similarity-based equivalence classes. In this paper we shall call them schemata (schema in singular form). If $\mathbf{h}$ is a schema in a

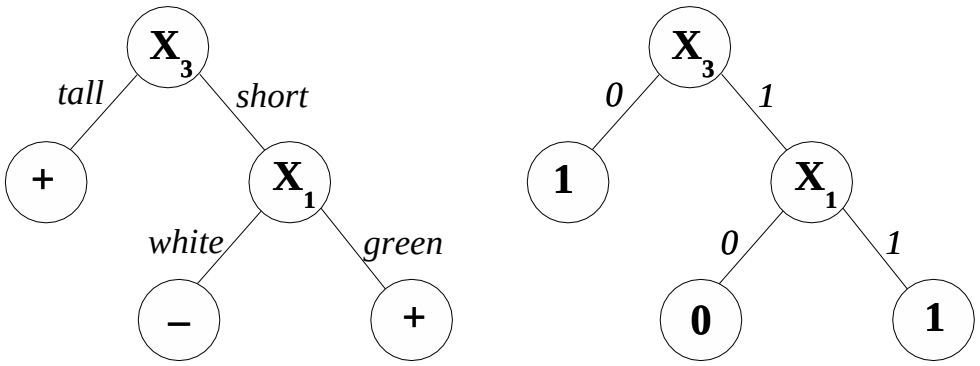

Figure 1: (Left) A symbolic decision tree and (Right) the numeric (Boolean) version that simply replaces the symbols by Boolean numbers.

Boolean domain, then we can represent it as $\mathbf{h} \in \{0, 1, *\}^\ell$, where $*$ denotes a wild-card character that matches any value of the corresponding feature. For example, consider a three-bit function defined by features x_1, x_2, and x_3, $f : \{0, 1\}^3 \rightarrow \{0, 1\}$. Let the tree shown in Figure 1(Right) be a representation of that function. The path $\{(x_3 \stackrel{1}{\rightarrow} x_1 \stackrel{0}{\rightarrow} f(\mathbf{x} = 0)\}$ in Figure 1(Right) 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 order of schema $\mathbf{h}$ is the number of non-wildcard values in $\mathbf{h}$. For example, the order of schema $0 * 1$ is two. A schema in a non-Boolean domain can be defined in a similar manner where the fixed values can be non-binary.

The Fourier spectrum of a decision tree can be easily computed by traversing the leaf nodes in a systematic fashion. Let Λ be the complete instance space of cardinality $|\Lambda|$. In case of Boolean feature space it is the set of all ℓ-bit binary strings. If the features are non-Boolean, Λ is the corresponding set of all ℓ-dimensional non-Boolean strings. Let us also assume that there are n leaf nodes in the decision tree and the i-th leaf node covers a subset of Λ, denoted by the schema $\mathbf{h}_{(i)}$. Therefore, $\cup_{i=1}^{n} \mathbf{h}_{(i)} = \Lambda$.

The $\mathbf{j}$-th coefficient of the spectrum can be computed as follows:

$$\begin{aligned} w_{\mathbf{j}} &= \frac{1}{|\Lambda|} \sum_{\mathbf{x} \in \Lambda} f(\mathbf{x}) \psi_{\mathbf{j}}(\mathbf{x}) \\ &= \frac{1}{|\Lambda|} \sum_{\mathbf{x} \in \mathbf{h}_{(1)}} f(\mathbf{x}) \psi_{\mathbf{j}}(\mathbf{x}) + \cdots + \frac{1}{|\Lambda|} \sum_{\mathbf{x} \in \mathbf{h}_{(n)}} f(\mathbf{x}) \psi_{\mathbf{j}}(\mathbf{x}) \end{aligned} \tag{2}$$

Now note that $f(\mathbf{x})$ takes a constant value for every member $\mathbf{x}$ in $\mathbf{h}_{(i)}$. Also, consider partition $\mathbf{j}$ whose all non-wildcard values are defined for attributes that are found in the path to $\mathbf{h}_{(i)}$. Then, since the path to a leaf node represents a schema, with some abuse of symbols we can write,

$$w_{\mathbf{j}} = \frac{|\mathbf{h}_{(1)}|}{|\Lambda|} f(\mathbf{h_{(1)}}) \psi_{\mathbf{j}}(\mathbf{h_{(1)}}) + \cdots + \frac{|\mathbf{h}_{(n)}|}{|\Lambda|} f(\mathbf{h_{(n)}}) \psi_{\mathbf{j}}(\mathbf{h_{(n)}}) \tag{3}$$

Where $f(\mathbf{h_{(i)}})$ is the class label of the leaf node i. For every path from the root to a leaf node (schema $\mathbf{h}$), all non-zero Fourier coefficients are detected by enumerating all possible values for each attribute in $\mathbf{h}$. The running time to compute a FC is $O(n)$, where n is the number of leaf nodes. We can therefore compute every coefficient in the spectrum by simply visiting the leaf nodes.

x_1	x_2	x_3	$f(\mathbf{x})$
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

FC	value
w_{000}	3/4
w_{001}	1/4
w_{010}	0
w_{011}	0
w_{100}	-1/4
w_{101}	-1/4
w_{110}	0
w_{111}	0

Table 1: (Left) The complete 3-bit domain of the decision tree in Figure 1. (Right) Its Fourier spectrum.

However, as we will see in the coming sections, we do not need to compute all the coefficients; only the low order coefficients are sufficient for most applications. Moreover, in practice the algorithm can benefit a lot from some pre-processing of the tree. Note that the computation of a FC (from Equation 3) requires the value of $|\mathbf{h}_{(i)}|$ and the corresponding class label. However, if the order of the coefficient is less than the depth of a leaf node then we can compute it without visiting all the leaf nodes, provided we maintain some statistics in the internal nodes. In order to illustrate the idea let us define the following quantity in a recursive manner.

$$\begin{aligned}\beta(v) &= |\mathbf{h}_{(v)}|\psi_{\mathbf{j}}(\mathbf{h}_{(v)}) \quad \text{if } v \text{ is a leaf node} \\ &= \beta(\text{LeftChild}(v)) + \beta(\text{RightChild}(v)) \quad \text{Otherwise}\end{aligned}$$

We can compute $\beta(v)$ for every internal node and store that information in the corresponding node. This statistics allows computation of low order FCs without visiting the leaf nodes every time. The proposed algorithm is fast and the overhead of computing these coefficients is minimal compared to the time needed for learning the tree.

Let us now illustrate the computation of Fourier coefficients from a tree and demonstrate that it is indeed the correct coefficient corresponding to the underlying function. Let us consider the decision tree in Figure 1 defined over 3-bit Boolean feature vectors. First, we use Equations 1 and 3, to illustrate the computation of the Fourier coefficients from the truth-table presented in Table 1 (Left). Then, we show how the coefficients can be computed from the decision tree.

Using Equation 1 we can write,

$$\begin{aligned}w_{000} &= \frac{1}{2^3}\sum_{\mathbf{x}} f(\mathbf{x})\psi_{000}(\mathbf{x}) = \frac{1+0+1+0+1+1+1+1}{8} = 3/4. \\ w_{001} &= \frac{1}{2^3}\sum_{\mathbf{x}} f(\mathbf{x})\psi_{001}(\mathbf{x}) = \frac{1+0+1+0+1+(-1)+1+(-1)}{8} = 1/4.\end{aligned}$$

Now using Equation 3,

$$\begin{aligned}w_{000} = &\ \tfrac{4}{8}f(**0)\psi_{000}(**0) + \tfrac{2}{8}f(0*1)\psi_{000}(0*1) + \tfrac{2}{8}f(1*1)\psi_{000}(1*1) = \tfrac{4}{8}+0+\tfrac{2}{8} = 3/4. \\ w_{001} = &\ \tfrac{4}{8}f(**0)\psi_{001}(**0) + \tfrac{2}{8}f(0*1)\psi_{001}(0*1) + \tfrac{2}{8}f(1*1)\psi_{001}(1*1) = \tfrac{4}{8}+0+-\tfrac{2}{8} = 1/4.\end{aligned}$$

Let us now consider a partition $\mathbf{j}$ that includes a non-wildcard value for attribute x_2. One such partition is 111. Note that x_2 does not appear in any of the paths in the tree. Then, let us see the computation of w_{111} using Equation 2,

$$
\begin{aligned}
w_{111} &= \frac{4}{8} f(**0)\psi_{111}(**0) + \frac{2}{8} f(0*1)\psi_{111}(0*1) + \frac{2}{8} f(1*1)\psi_{111}(1*1) \\
&= \frac{2}{8} f(**0)(\psi_{111}(000) + \psi_{111}(010) + \psi_{111}(100) + \psi_{111}(110)) + \\
&\quad \frac{2}{8} f(0*1)(\psi_{111}(001) + \psi_{111}(011)) + \frac{2}{8} f(1*1)(\psi_{111}(101) + \psi_{111}(111)) \\
&= \frac{2}{8} f(**0)(1 + (-1)) + \frac{2}{8} f(0*1)(1 + (-1)) + \frac{2}{8} f(1*1)(1 + (-1)) \\
&= 0 + 0 + 0 = 0.
\end{aligned}
$$

Note that $w_{111} = 0$ since no path in the tree contains x_2. So far our discussion pointed out the following things:

1. A decision tree can be viewed as a discrete function. If it is a symbolic function then we can maintain a small code-book and convert the symbolic tree to a numeric one.

2. We can efficiently compute the Fourier spectrum of the numeric decision tree.

However, we still have not discussed why we should compute the Fourier spectrum of a decision tree. It turns out that the Fourier representation of a decision tree with bounded depth has some very interesting properties [22, 24, 29]. These observations are discussed in the following section.

2.3 Fourier Spectrum of a Decision Tree: Why Bother?

For almost all practical applications decision trees have bounded depths. The Fourier spectrum of a bounded depth decision tree has some interesting properties that are listed below.

1. The Fourier representation of a bounded depth (say k) decision tree has only a polynomial number (polynomial in the number of features defining the domain) of non-zero coefficients; all coefficients corresponding to partitions involving more than k feature variables are zero. As we saw in the previous section, if partition $\mathbf{j}$ contains a non-zero value for x_k that is not defined in path $\mathbf{h}$, $\sum_{\mathbf{x} \in \mathbf{h}} f(\mathbf{x})\psi_{\mathbf{j}}(\mathbf{x}) = 0$. Since k is the maximum order of all possible $\mathbf{h}$ in a decision tree, for any partition $\mathbf{j}$ whose order is greater than k, $w_{\mathbf{j}}$ is zero.

2. 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. This was proved in [24] for Boolean decision trees. Its counterpart for trees with non-boolean features can be found elsewhere [29, 28].

These observations point out that the spectrum of the decision tree can be approximated by computing only a small number of low-order coefficients. So Fourier bases offer an efficient numeric representation of a decision tree in the form of an algebraic function that can be easily stored, transmitted, and manipulated.

Fourier spectrum of a decision tree tells us more than the interactions among the features. These coefficients can also tell us about the distribution of class labels at any node of the decision tree. This property can in fact be exploited for constructing a tree from the spectrum. The following section explores this.

2.4 From Fourier Spectrum To Decision Trees

The distribution of class labels in a schema is an important aspect since that identifies the utility of the schema as a decision rule. For example, if all the members of some schema $\mathbf{h}$ has a class label value of 1 then $\mathbf{h}$ can be used as an effective decision rule. On the other hand, if the proportion of label values 1 and 0 is almost equal then $\mathbf{h}$ cannot be used as an effective decision rule. In decision tree learning algorithms like ID3 and C4.5 this "skewness" in the distribution for a given schema is measured by computing the *information-gain* measure defined in the following. The information gain resulted from fixing the feature x_i in schema $\mathbf{h}$ is,

$$\text{Gain}(\mathbf{h}, x_i) = \text{Entropy}(\mathbf{h}) - \sum_{v \in \{0,1,\cdots \lambda_i - 1\}} \frac{|\mathbf{h}_{\langle x_i = v \rangle}|}{|\mathbf{h}|} \text{Entropy}(\mathbf{h}_{\langle x_i = v \rangle})$$

where $\mathbf{h}_{\langle x_i = v \rangle}$ represents the schema obtained by replacing the wild-card character at the i-th position of $\mathbf{h}$ with v.

For Boolean classification (two class problem), entropy of some schema $\mathbf{h}$ is defined in terms of the schema average function $\phi(\mathbf{h})$:

$$\begin{aligned} \phi(\mathbf{h}) &= \frac{1}{|\mathbf{h}|} \sum_{\mathbf{x} \in \mathbf{h}} f(\mathbf{x}) \qquad (4) \\ \text{Entropy}(\mathbf{h}) &= -\phi(\mathbf{h}) \log \phi(\mathbf{h}) - (1 - \phi(\mathbf{h})) \log(1 - \phi(\mathbf{h})) \end{aligned}$$

where $f(\mathbf{x})$ is the classification value of $\mathbf{x}$.

The computation of $\phi(\mathbf{h})$ using Equation 4 is not practically feasible, since we need to evaluate all $\mathbf{x} \in \mathbf{h}$. Instead, following [15, 19] we can compute $\phi(\mathbf{h})$ directly from the given Fourier spectrum:

$$\phi(\mathbf{h}) = \sum_{l_1} .. \sum_{l_m} \exp^{2\pi i ((\frac{l_1 b_1}{\lambda_{j_1}} + .. + \frac{l_m b_m}{\lambda_{j_m}})} w_{(0,..,l_1,..,l_m,..,0)} \qquad (5)$$

where $\mathbf{h}$ has m non-wildcard value b_i at position j_i and l_i has the cardinality of λ_i.

Equation 5 can be directly used to construct a C4.5-style algorithm to construct a decision tree from the spectrum. However, such a naive approach is computationally inefficient. The computation of $\phi(\mathbf{h})$ requires an exponential number of FCs with respect to the order of $\mathbf{h}$. Thus, the cost involved in computing $\phi(\mathbf{h})$ increases exponentially as the tree becomes deeper. Moreover, since the Fourier spectrum of a decision tree is very compact in size, most FCs involved in computing $\phi(\mathbf{h})$ are zero. Therefore, the evaluation of $\phi(\mathbf{h})$ using Equation 5 is not only inefficient but also involves unnecessary computations.

An efficient algorithm that computes $\phi(\mathbf{h})$ can be constructed by taking advantage of sparseness of the Fourier spectrum and decomposable property of Equation 5. When computing the average of order l schema $\mathbf{h}$, we can reduce some computational steps if any of order l-1 schemata which subsumes $\mathbf{h}$ is already evaluated. For a simple example in a 6-bit Boolean domain, let us consider the evaluation of schema $\phi(*1*0**)$. Let us also assume that $\phi(*1****)$ is pre-calculated. Then, $\phi(*1*0**)$ is computed by simply adding w_{000100} and $-w_{010100}$ to $\phi(*1****)$. Recall that a path (from the root node) to any node is represented as a schema. Then, given a path $\mathbf{h}$ of order k, choosing an attribute for the next node is essentially the same as selecting the best schema among those *candidate* schemata of order k+1 which are subsumed by $\mathbf{h}$. Therefore, computing the averages of all candidate schemata by scanning over the Fourier spectrum makes the entire decision tree induction process become very efficient. The Tree Construction from Fourier Spectrum (TCFS) algorithm that is based on this observation is detailed in [31, 30].

This section described that a Fourier spectrum can be effectively used to construct a decision tree. The following section considers the problem of aggregating multiple decision trees in a ensemble classifier using their Fourier spectrum.

3 Aggregating Multiple Trees

The Fourier spectrum of a decision tree-ensemble classifier can also be computed using the algorithm described in the previous section. We first have to compute the Fourier spectrum of every tree and then aggregate them using the chosen scheme for constructing the ensemble. Let $f_e(\mathbf{x})$ be an ensemble of m different decision trees where the output is a weighted linear combination of the outputs of these trees. We consider weighted linear combination since most of the existing schemes for ensemble learning takes that approach.

$$\begin{aligned} f_e(\mathbf{x}) &= c_1 f_1(\mathbf{x}) + c_2 f_2(\mathbf{x}) + ... + c_n f_m(\mathbf{x}) \\ &= c_1 \sum_{j \in Z_1} w_{\mathbf{j}}^{(1)} \overline{\psi_{\mathbf{j}}}(\mathbf{x}) + \cdots + c_n \sum_{j \in Z_m} w_{\mathbf{j}}^{(m)} \overline{\psi_{\mathbf{j}}}(\mathbf{x}) \end{aligned}$$

where $f_i(\mathbf{x})$ and c_i are i^{th} decision tree and its weight in the ensemble respectively. Z_i is the set of all partitions with non-zero Fourier coefficients in the spectrum of the i^{th} decision tree and $w_{\mathbf{j}}^{(i)}$ is a Fourier coefficient in that spectrum. Now we can write,

$$f_e(\mathbf{x}) = \sum_{j \in Z} w_{\mathbf{j}} \overline{\psi_{\mathbf{j}}}(\mathbf{x}) \tag{6}$$

where $w_{\mathbf{j}} = \sum_{i=1}^{m} c_i w_{\mathbf{j}}^{(i)}$ and $Z = \cup_{i=1}^{m} Z_i$. Therefore, the Fourier spectrum of $f_e(\mathbf{x})$ (an ensemble classifier) is simply the weighted sum of the spectra of the member trees.

The spectrum of an ensemble of trees can be directly useful for data mining in different application domains. Mining data streams, distributed data mining, building highly accurate classifiers are some examples. In the rest of this paper we focus on mining financial data streams in a mobile environment where the Fourier spectrum offers the following immediate benefits:

1. aggregation of trees generated from different blocks of data observed at different time from data streams,
2. visualization of the ensemble through its Fourier spectrum,
3. minimizing the communication overhead,

The following section briefly describes a mobile data mining system that deploys the Fourier representation-based approach described in this paper. Although all the different modules of the MobiMine are not directly relevant to the techniques presented so far in this paper, we believe that this description of the application is useful for appreciating the motivation and the contribution of the current work in real-life problem solving.

4 The MobiMine System

This section presents a brief description of MobiMine, a mobile application that exploits the Fourier representation of decision trees reported in this paper. The overview presented in this section covers

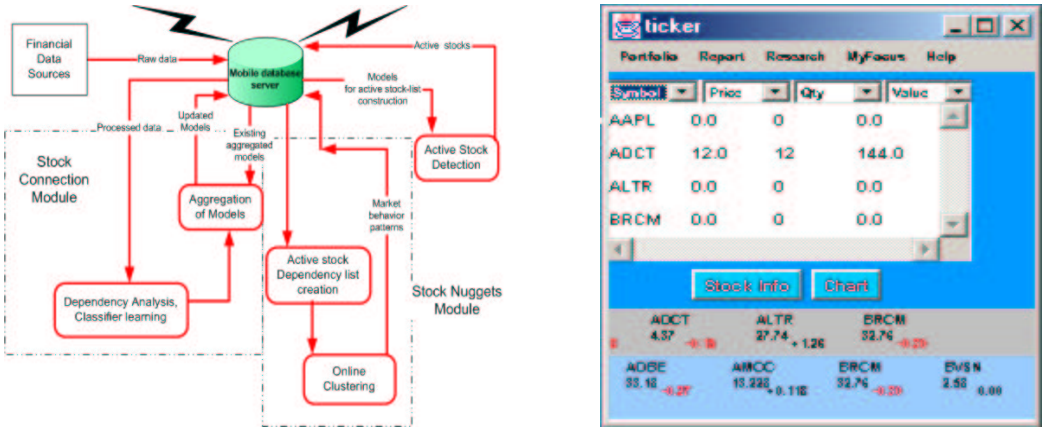

Figure 2: (Left) The architecture of the MobiMine Server. (Right) The main interface of MobiMine. The bottom-most ticker shows the WatchList; the ticker right above the WatchList shows the stocks in the portfolio.

the different modules of the MobiMine; not all of them make use of the techniques discussed so far in this paper. However, a general description is necessary to cast the current contribution in the context of a real application environment.

4.1 An Overview of the System

The MobiMine is a client-server application. The clients (Figure 3), running on mobile devices like hand-held PDAs and cell-phones, monitor a stream of financial data coming through the MobiMine server (Figure 2 (Left)). The system is designed for currently available low-bandwidth wireless connections between the client and the server. In addition to different standard portfolio management operations, the MobiMine server and client apply several data mining techniques in order to offer the user a variety of different tools for monitoring the stock market at any time from any where. Figure 2(Right) shows the main user interface of the MobiMine.

The main functionalities of the MobiMine are listed in the following:

1. Portfolio Management and Stock Tickers: Standard book-keeping operations on stock portfolios including stock tickers to keep an eye on the performance of the stocks in the portfolio.

2. FocusArea: Stock market data is often overwhelming. It is very difficult to keep track of all the developments in the market. Even for a full-time professional following the developments all the time is challenging. It is undoubtedly more difficult for a mobile user who is likely to be busy with other things. The MobiMine system offers a unique way to monitor changes in the market data by selecting a subset of the events that is more "interesting" to the user. This is called the FocusArea of the user. It is a time varying feature and it is currently designed to support the following functionalities:

   (a) WatchList: The system applies different measures to assign a score to every stock under observation. The score is an indication of the "interesting-ness" of the stock. A relatively higher score corresponds to a more interesting stock. A selected bunch of "interesting" stocks goes through a personalization module in the client device before it is presented to the user in the form of a WatchList.

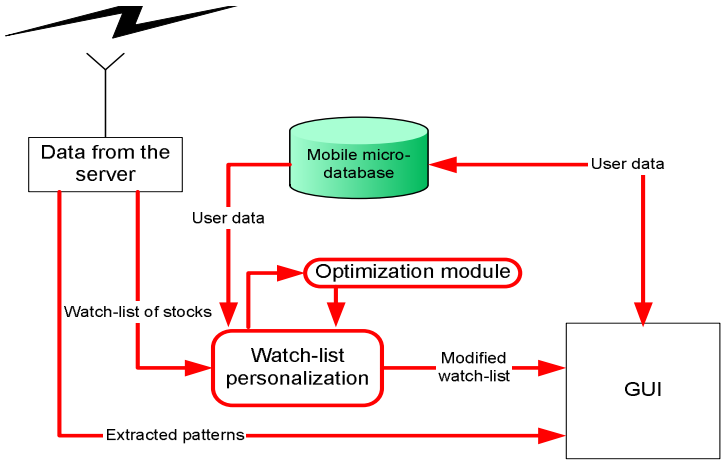

Figure 3: The architecture of the MobiMine Client.

(b) Context Module: This module offers a collection of different services for better understanding of the time-critical dynamics of the market. The main interesting components are,

i. StockConnection Module: This module allows the user to graphically visualize the "influence" of the currently "active" stocks on the user's portfolio. This module detects the highly active stocks in the market and presents the causal relationship between these and the stocks in user's portfolio, if any. The objective is to give the user a high level qualitative idea about the possible influence on the portfolio stocks by the emerging market dynamics.

ii. StockNuggets Module: The MobiMine Server continuously processes a data stream defined by a large number of stock features (fundamentals, technical features, evaluation of a large number of well-known portfolio managers). This module applies on-line clustering algorithms on the active stocks and the stocks that are usually influenced by them (excluding the stocks in the user's portfolio) in order to identify similarly behaving stocks in a specific sector.

The StockConnection module tries to detect the effect of the market activity on user's portfolio. On the other hand, the StockNuggets module offers an advanced stock-screener-like service that is restricted to only time-critical emerging behavior of stocks.

(c) Reporting Module: This module supports a multi-media based reporting system. It can be invoked from all the interfaces of the system. It allows the user to watch different visualization modules and record audio clips. The interface can also invoke the e-mail system for enclosing the audio clips and reports.

A detailed description of this system can be found elsewhere [20]. The following section discusses the unique philosophical differences between the MobiMine and traditional systems for mining stock data.

4.2 MobiMine: What It is Not

A large body of work addresses different aspects of stock forecasting [2, 3, 9, 21, 23, 42] and selection [8, 18] problem. The MobiMine is fundamentally different from most of these systems for stock

forecasting and selection. First of all, it is different on the basis of philosophical point of view. In a traditional stock selection or portfolio management system the user initiates the session. User outlines some preferences and then the system looks for a set of stocks that satisfy the constraints and maximizes some objective function (e.g. maximizing return, minimizing risk). The MobiMine does not do that. Instead, it initiates an action, triggered by some activities in the market. The goal is to draw user's attention to possibly time-critical information. For example, if the Intel stock is under-priced but its long time outlook looks very good then a good stock selection system is likely to detect Intel as a good buy. However, the MobiMine is unlikely to pick Intel in the WatchList unless Intel stock happens to be highly active in the market and it fits with user's personal style of investment. The Context detection module is also unlikely to show Intel in its radar screen unless Intel happens to be highly influenced by some of the highly active stocks in the market. This difference in the design objective is mainly based on our belief that mobile data mining systems are likely to be appropriate only for time-critical data. If the data is not changing right now, probably you can wait and you do not need to keep an eye on the stock price while you are having a lunch with your colleagues.

Most of the stock selection systems are based on predictive models. There exists a large body of work that approaches this problem using statistical techniques, neural networks, genetic algorithms, and other techniques. The MobiMine in its current design does not perform any kind of prediction of future behavior. The StockConnection module presents the influence diagram based on the current data. It simply presents how the different companies have been influencing the portfolio stocks in the recent past. As we noted earlier, the StockNuggets module is a clustering-based screener for the active stocks and those (excluding the portfolio stocks) strongly influenced by the active stocks. This is another major distinction between the traditional stock selection systems and the MobiMine. In short, the MobiMine system is designed based on a minimalist principle—anything that can wait should wait and therefore the MobiMine does not need to support it. It should only support functionalities required for time-critical events. The following section identifies the role of decision tree ensembles in monitoring stock market activities using the MobiMine.

4.3 Portfolio and Market Activities

Understanding the interaction between a portfolio and the market is a non-trivial problem. It often requires extensive data analysis, in depth understanding of the driving factors in the market, and years of experience in the stock market. In fact, PDA may not be the right kind of platform to perform extensive user mediated market and portfolio analysis. The MobiMine does not make any attempt to do that. Its StockConnection module offers short-term dependency among a pair of stocks based on the data from the recent past. This does not tell the user about what will happen to the dependent stock in the near future. It simply suggests the user to keep an eye on a stock since it is statistically correlated or influenced by a currently active stock. The MobiMine employs statistical, Bayesian, and decision tree-based on-line techniques to detect the dependencies.

Detecting dependencies among different stocks and the corresponding *technical* features in an on-line fashion requires algorithms that can either update the models incrementally or work by adding new models to an ensemble. Although MobiMine makes use of a collection of different on-line techniques, this section will focus only on the decision tree ensembles.

There exist several known techniques to construct incremental decision tree-based models. Some of the earlier efforts include ID4[37], ID5[39], ID5R[40] and ITI[41]. All these systems work using the ID3 style "information gain" measure to select attributes (or, equivalently, decision nodes). They are all designed to incrementally build a decision tree using one training instance at a time by keeping necessary statistics (measure for information gain) at each decision node. Some of the

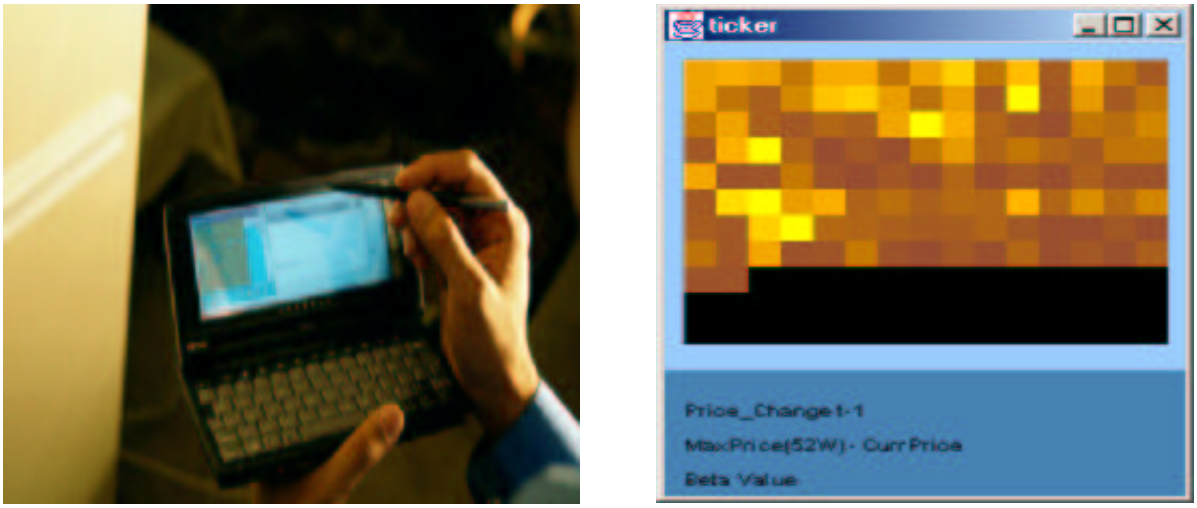

Figure 4: (Left) Fourier spectrum-based decision tree mining interface for HP Jornada. (Right) Fourier spectrum-based decision tree visualization module in MobiMine. The screen shows the influence of different market features on a stock. The interface plots the coefficients of the Fourier Spectrum of the decision tree ensemble. Each cell corresponds to a unique coefficient and the cells are color-coded based on their magnitudes. Every cell corresponds to a unique subset of features. Brighter cells correspond to influencing features. The bottom window lists these features.

recent techniques designed for large-scale applications include the Bootstrapping-based BOAT [14] and the Hoeffding (additive Chernoff bound) [16] tree-based VFDT [10] and the CVFDT [17].

An ensemble-based approach is proposed in [12]. It works using a Boosting-based approach to create ensemble of models. Different trees are generated from different blocks of data observed at different time intervals. The ensemble classifier for the stream is defined by a weighted average of the outputs of these trees. Breiman proposed an arcing method for learning models from large data sets and stream data [5, 6]. It is also based on adaptive re-sampling. However, it uses unweighted average to build the ensemble classifier. Recently Street and Kim [38] proposed Streaming Ensemble Algorithm (SEA) that learns an ensemble of decision trees for large-scale classification. SEA maintains a fixed number of classifiers. Once the ensemble becomes full, the k-th classifier C_k is added only if it outperforms any previous C_i in the ensemble. In that case, C_i is removed from the ensemble. Performance is measured using the most current data block.

Both the ensemble-based and incremental techniques in practice generate a collection of decision trees. In order to provide the user with the explanatory market models we need to send the trees to the client devices over the wireless network and present them in a user-friendly format. The MobiMine makes use of the Fourier representation for aggregating the ensemble of decision trees in a compressed representation that offers a compact but meaningful way to visualize in a PDA. The Fourier representation also offers an efficient way to transmit the trees over the wireless network.

The accuracy of the aggregation of trees and their transmission from the server to the client will be studied in details later in this paper. First, let us briefly discuss its simple but effective Fourier spectrum-based visualization module for detecting dependencies among stocks and their different characteristic features.

The Context module of the MobiMine offers an interface to visualize decision trees through their Fourier spectrum. The interface is interactive and it does not require any prior knowledge of Fourier

analysis. Figure 4 shows a typical session of this interface. It shows the Fourier coefficients of an ensemble of decision trees. Each rectangular cell on the interface corresponds to a unique coefficient. Each cell is color-coded according to the magnitude (or significance) of the corresponding Fourier coefficient. When the user clicks on a significant (or brighter) cell, all the features associated with the corresponding Fourier coefficient are displayed; these features represent the dominant ones that significantly influence the class label. This module can help user identifying the stocks and features that are strongly influencing the observed class label values (fluctuations in the portfolio stocks in the current application). The following section presents the experimental results regarding aggregation and transmission of decision trees.

5 Mining Financial Data Streams

This section presents the results of several experiments that documents the empirical characteristics of the Fourier representation of decision trees. It reports the accuracy of the Fourier representation of decision trees using different degrees of approximation. It also reports the accuracy of the proposed aggregation approach for combining multiple decision trees using Bagging and Arcing. Finally it presents experimental results documenting the cost of transmitting the Fourier spectrum of decision trees over a wireless network.

In order to study the properties of the proposed technique in a systematic manner, we performed controlled experiments using a semi-synthetic data stream with 174 Boolean attributes. The objective is to continuously evolve a decision tree-based model for a Boolean attribute. The data-stream generator is essentially a C4.5 decision tree learned from three years of S&P100 and Nasdaq 100 stock quote data. The original data is pre-processed and transformed to discrete data by assigning discrete values for "increase" and "decrease" in stock quote between consecutive days (i.e. local gradient). Decision trees classify the ups-and-downs of the Yahoo stock based on the attribute values of the 174 stocks.

We assume a non-stationary sampling strategy in order to generate the data. Every leaf in the decision tree-based data generator is associated with a certain probability value. Data samples are generated by choosing the leaves according to the assigned probability distribution. This distribution is changed several times during a single experiment. We also added white noise to the generator. Test data set is comprised of $10,000$ instances. The following sections report the experimental results.

5.1 Accuracy of Fourier Aggregated Ensembles

This section considers several popular ensemble learning techniques and demonstrates that the Fourier spectrum-based approach can be used for accurately aggregating multiple decision trees.

As noted earlier, the ensemble learning literature offers different ways to construct classifiers. Most of these techniques compute the outputs of the member classifiers of the ensemble and produces an overall output by taking their weighted average.

Averaging the outputs of the individual models with uniform weight is probably the simplest possibility. This technique [32, 27] is called the Basic Ensemble Method (BEM) or naive Bagging. Breiman proposed an Arcing method Arc-fx [4, 5] for creating ensembles. It is fundamentally based on the idea of Arcing – adaptive re-sampling by giving higher weights to those instances that are usually mis-classified. We consider both of these ensemble learning techniques to create an ensemble of decision trees from the data stream. These trees are however combined using the proposed Fourier spectrum-based approach which the regular ensemble learning techniques do not offer. We also performed experiments using an AdaBoost-based approach suggested elsewhere [12].

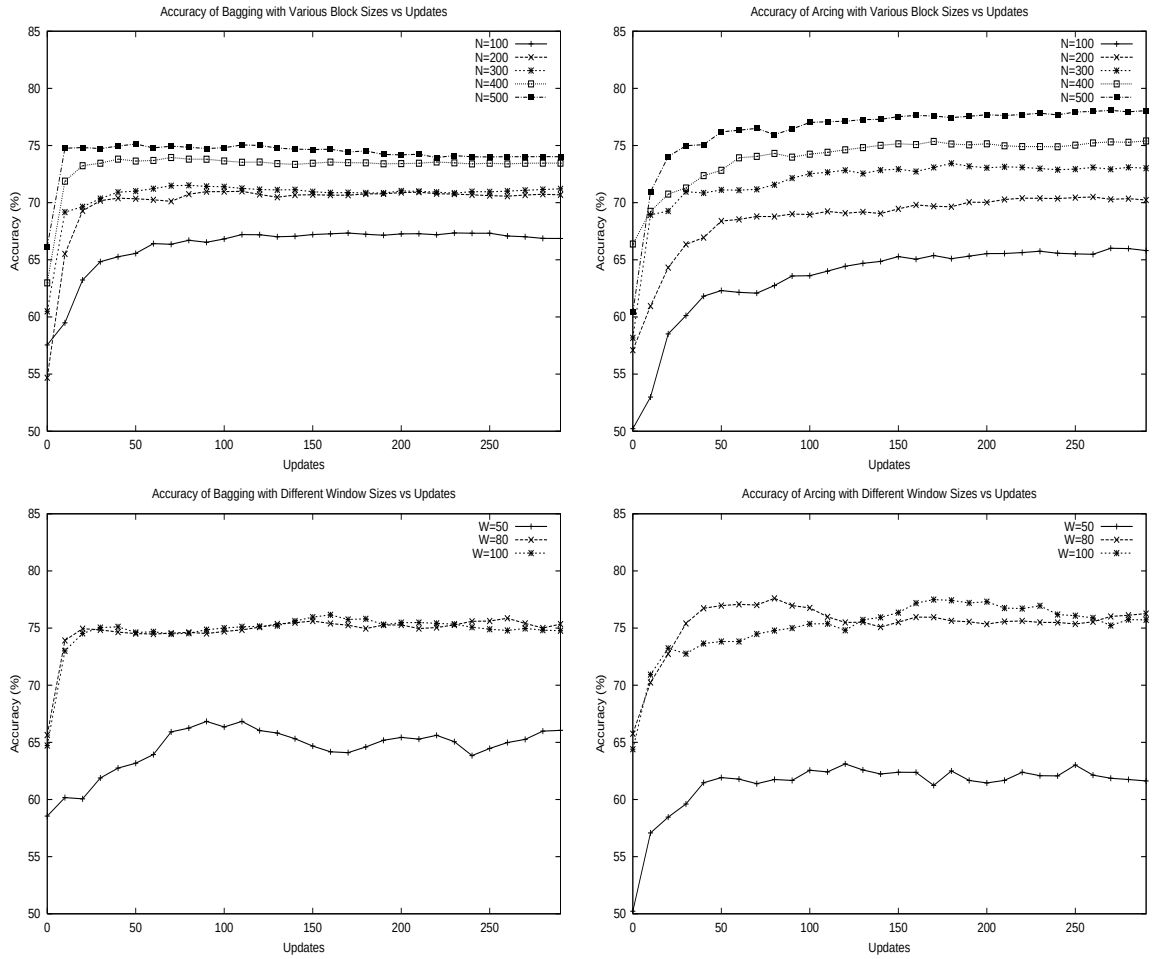

Figure 5: (Top) Accuracy vs. iterations with various data block sizes (N): (Left) Bagging (Right) Arcing. (Bottom) Accuracy vs. iterations with various window sizes (W): (Left) Bagging (Right) Arcing.

However, we choose not to report that since its performance appears to be considerably inferior to those of Bagging and Arcing for the data set we used.

Not all the models generated from different data blocks should always be aggregated together. Sometimes we may want to use a pruning algorithm [26, 33] for selecting the right subset of models. Sometimes a "windowing scheme" [12, 5] can be used where only W most recent classifiers are used for learning and classification. Our experiments consider some of these possibilities too.

We implemented the naive Bagging and Arcing techniques and performed various tests over the validation data set as described below. Decision tree models are generated from every data block collected from the stream and their spectra are combined using the BEM and Arcing. We studied the accuracy of each model with various sizes (N) of data block at each update. We use $N = 100, 200, 300, 400, 500$. We also studied the accuracy of each model generated using the "windowing" technique with various window sizes, $W = 50, 80, 100$. All the results are measured over 300 iterations where every iteration corresponds to a unique discrete time step.

Figure 5 (Top) plots the classification accuracies of Bagging and Arcing with various data block sizes. Bagging converges rapidly with all different block sizes before or around 50 iterations,

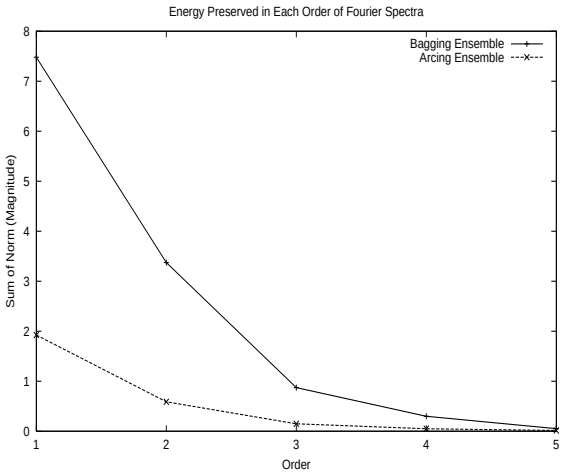

Figure 6: Energy preserved in each order of Fourier Spectra. Each graph shows energy distribution of ensemble of 300 decision trees for Bagging and Arcing.

while Arcing shows gradual increase throughout all iteration steps. Figure 5 (Bottom) plots the classification accuracies of Bagging and Arcing with various window size. With relatively large window size (80 and 100), we observed small decrease in accuracies in both cases.

5.2 Approximating the Fourier Spectrum

This section reports experimental results to demonstrate that the Fourier spectrum of decision tree ensembles can be accurately approximated by a relatively small number of low order Fourier coefficients.

Let us first define a few symbols for representing the quantities that we measure in the experiments. Let $\mathcal{F}_k$ be the set of all Fourier coefficients of order k, as defined in the following:

$$\mathcal{F}_k \quad = \quad \{w_{\mathbf{j}} \mid \text{order}(\mathbf{j}) = k,\ w_{\mathbf{j}} \neq 0\} \tag{7}$$

where order($\mathbf{j}$) denotes the order of the partition $\mathbf{j}$. The *energy* of the set $\mathcal{F}_k$ is

$$E(\mathcal{F}_k) \quad = \quad \sum_{w_{\mathbf{j}} \in \mathcal{F}_k} ||w_{\mathbf{j}}||^2 \tag{8}$$

where $||w_{\mathbf{j}}||$ returns the magnitude of the coefficient. Note that the Fourier Coefficient can be a complex number in the general case with non-Boolean categorical attributes.

Figure 6 shows the variation of $E(\mathcal{F}_k)$ with respect to k for both Bagging and Arcing ensemble models discussed in the previous section. The maximum depth of all the decision trees in the ensemble is ten and that bounds the maximum order of any Fourier coefficient to ten. However, as clearly demonstrated in the figure, most of the information (or energy) is captured by the low order coefficients. The graph also shows rapid decrease in energy as the order grows. It clearly demonstrates the exponential decay property noted earlier in this paper.

To see how the energy, preserved in each $\mathcal{F}_k$, affects the classification, we first performed classification accuracy test using the first two $\mathcal{F}_k$-s. We repeated the test by including the next higher $\mathcal{F}_{k+1}$ at a time. Next we study the effect of approximation on the accuracy of the classifier. We consider all the coefficients in $\cup_{i=0}^{k} \mathcal{F}_k$ and compute the corresponding accuracy in classification.

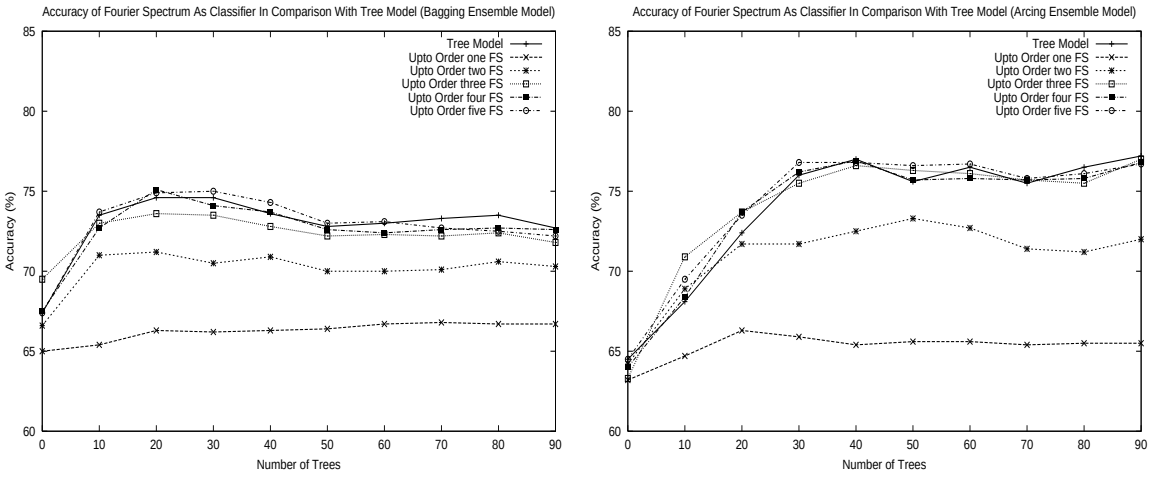

Figure 7: Classification accuracy of Fourier spectra as classifier when FC order is bounded (Left) Bagging, (Right) Arcing.

Model	Number of FCs	Number of Decision Nodes
Bagging	1,303	11,566
Arcing	1,752	13,426

Table 2: The number of FCs sufficient for classification, when compared to the number of decision nodes in the original ensemble model. Both FCs and decision nodes are collected from 100 decision trees.

Figure 7 reports that for different choices of k using Bagging and Arcing. This experiment was conducted in a way similar to the previous experiments with different number of data updates and validation data (100 and 5,000 each). Also it was restricted to a data block size of 500; the windowing scheme was not used since it is not relevant to the objective of this particular experiment. The experimental results indicate that approximations of the Fourier spectrum using low order coefficients are sufficient for good classification accuracy, comparable to the original tree-ensemble.

We also studied the size of Fourier spectrum of ensemble models in terms of the number of FCs since space complexity is an important issue. The size of the Fourier Spectrum can be significantly reduced by throwing away the FCs with small magnitude. In order to do this, we simply sorted up to order-5 FCs in the descending order by their magnitude and plotted the cumulative energy distribution. Figure 8 shows these graphs. For both Bagging and Arcing models, most energy are preserved in a small percentage of FCs, which indicates that most FCs have small magnitudes. It should be noted that percentages are defined over FCs we extracted from 100 trees, not over the entire set of all possible FCs. We also plotted classification accuracies versus percentage of FCs (from the top) and observed that less than one percentage of FCs is sufficient for classification (See Figure 9). In our implementation, 28 bytes are needed to hold a FC and 72 bytes for a decision node. When compared to the number of decision nodes in 100 decision trees, one percentage of FCs is significant reduction in representing an ensemble model. Table 2 compares the number of nodes in ensemble of decision trees and the number of FCs in the spectrum for approximately same accuracy of the classifiers.

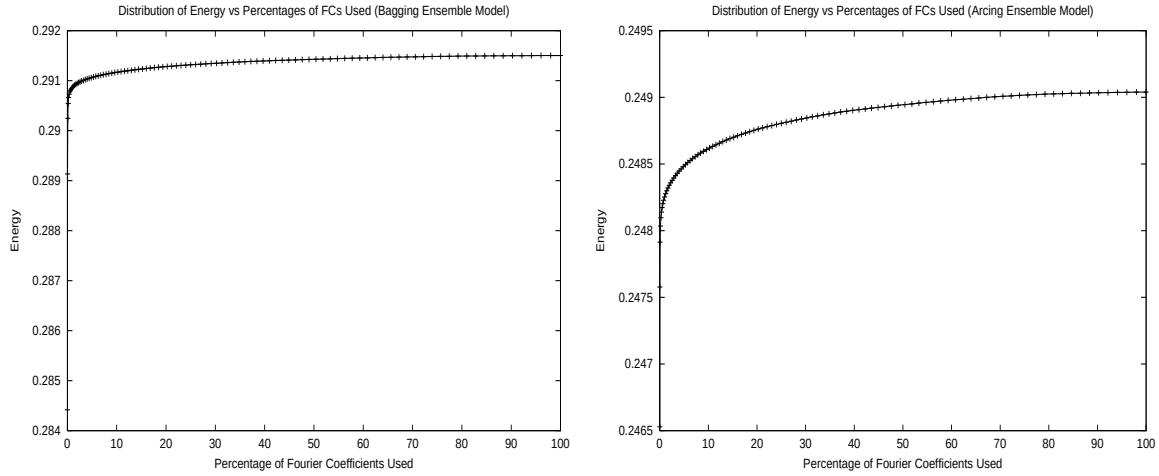

Figure 8: Distribution of energy vs percentage of FCs used in the approximation. from the sorted collection $\{\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \mathcal{F}_4, \mathcal{F}_5\}$ (Left) Bagging, (Right) Arcing.

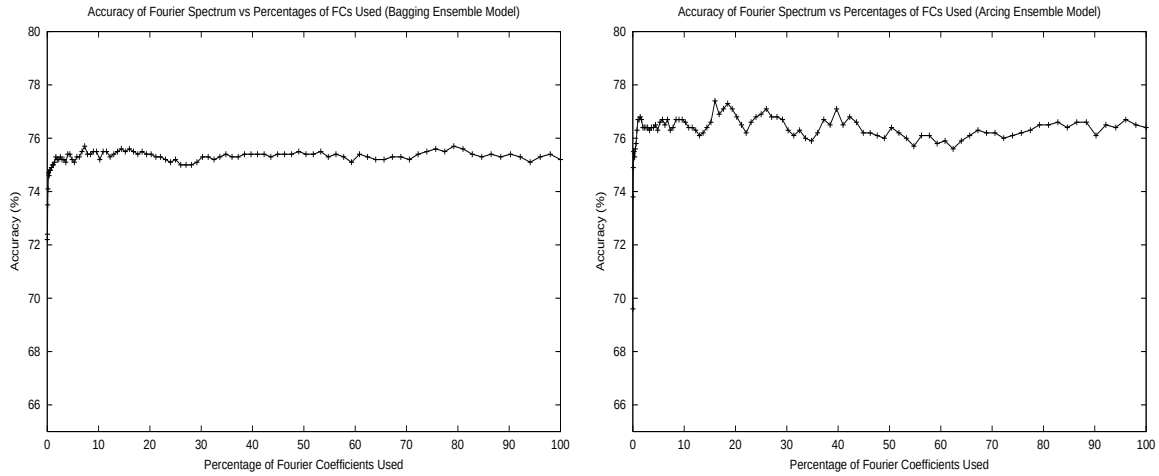

Figure 9: Classification accuracy vs percentage of FCs from the sorted $\{\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \mathcal{F}_4, \mathcal{F}_5\}$ (Left) Bagging, (Right) Arcing.

5.3 Transmission of Decision Trees in a Wireless Network

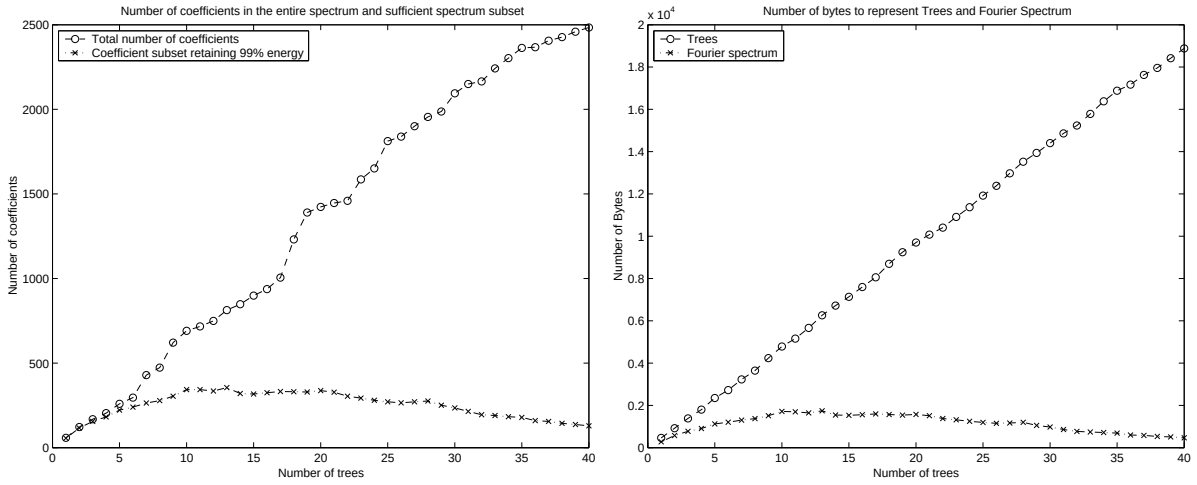

Figure 10: (Left) Variation of the total number of coefficients and significant coefficients in the Fourier spectrum. (Right) Byte size of the tree ensemble and its spectrum.

Mining data streams from a mobile device requires paying attention to the cost of communication. The ensemble-based decision tree learning techniques produce a stream of trees. Transmission of these trees over the low-bandwidth channel is difficult and expensive. So MobiMine sends the Fourier spectrum of the trees in order to reduce the transmission cost and improve the response time. The following part of this section offers experimental results that demonstrate the benefits of the Fourier representation of decision trees for their transmission over a wireless network.

The experiments use a stream of Nasdaq 100 stock data. At every time interval the data collection module gathers 100 different rows for 100 different companies. Data is collected every five minutes. For this experiments we used 45 numeric feature values. These features are preprocessed and discretized based on the percentages of difference between two consecutive instances. If the percentage of difference is beyond a certain threshold, the feature value is set to one, otherwise zero.

For this experiment, we built an ensemble of decision trees after 65 data updates, which is roughly one-day-data. The results we present in this section were performed using the trees generated for in the "computer sector" in Nasdaq 100 index. The sector includes 40 companies (40 trees). We mapped each decision tree to its corresponding Fourier spectrum and measured:

- The total size of both the aggregated Fourier spectrum and sufficient spectrum subset that holds most of energy.
- The number of bytes required to represent Fourier spectrum and the original ensemble.
- The time required to send Fourier spectrum and the original ensemble to a client.

All the experiments were conducted to verify that how both the original ensemble model and its Fourier spectrum scale up with larger number of trees in a mobile environment. In order to do this, we created ensembles of different sizes (the number of decision trees it contains).

We first performed a set of experiments in order to document the characteristics of the spectrum of the ensembles in a way described in the previous section. We sorted Fourier coefficients by their

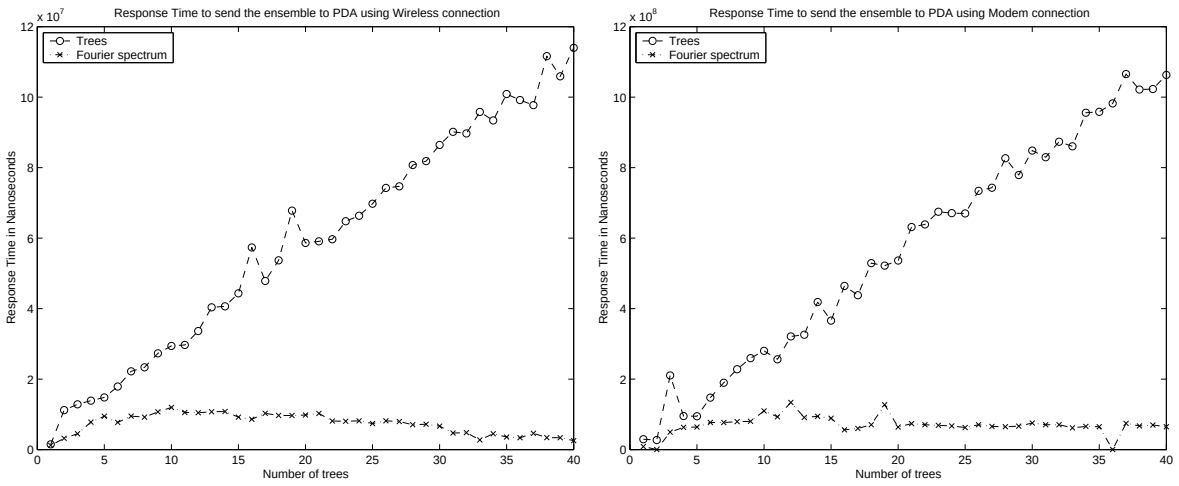

Figure 11: Variation of the response time for transmitting the trees and that of transmitting the *average* Fourier spectrum of the ensemble with respect to an increasing number of trees in the ensemble. (Left) Results using an IEEE 802.11b wireless channel. (Right) Results using a modem channel.

magnitudes and counted the number of coefficients (from the largest to the lowest) that contains 99% of the energy. Figure 10(Left) reports the result. This again indicates that a very small number of FCs is sufficient to represent an ensemble regardless of the number of trees it contains. In other words, even for a large ensemble, its Fourier spectrum remained very compact for the data set used here. Figure 10(Right) compares the overheads in storing a Fourier spectrum in the system with that of the original ensemble model.

In order to measure the time required to send the Fourier spectrum and the original ensemble to a client, we sent each model to a Compaq iPaq H3650 over an IEEE 802.11b wireless channel. Figure 11(Left) shows the average result of 50 independent runs. The graph indicates that we achieve a significant reduction in transmission time with the Fourier representation. The same set of experiment was conducted over a low bandwidth (56K) dial-up connection and reported in Figure 11(Right). With the decreased bandwidth, the difference in time for sending the ensemble and its Fourier spectrum increases and that makes the Fourier based approach even more attractive.

We would also like to point out that the Detection and extraction of significant FCs from a decision tree is very fast; we observed that it takes less than a second to extract significant FCs from a decision tree of around 150 nodes in a Pentium III 1 Giga Hz PC.

6 Future Work and Conclusion

The emerging domain of wireless computing is alluding the possibility of making data mining ubiquitous. However, this new breed of applications is likely to have different objectives and they will have to work with different system resource requirements. This will demand dramatic changes in the current desktop technology for data mining. This paper considered a small but important aspect of this issue.

This paper presented a novel Fourier analysis-based approach to enhance interaction with decision trees in a mobile environment. It observed that a decision tree is a function and its numeric functional representation in Fourier basis has several utilities; the representation is efficient and easy

to compute. It is also suitable for aggregation of multiple trees frequently generated by ensemble-based data stream mining techniques like the Boosting and Bagging. This approach also offers a new way to visualize decision trees that is completely different from the traditional tree-based presentation used in most data mining software. This paper also introduced the MobiMine, a mobile data mining system for assisted monitoring data streams from the financial market.

We are currently incorporating several additional data mining-based functionailities into MobiMine. We are also exploring several fundamental issues in mobile data mining. Some of them are listed below:

- Power consumption issues: Consumption of battery power is a serious issue in a PDA or other mobile device. The client system is currently being extensively tested for power consumption characteristics detection. Optimized modules of client systems are likely to decrease the power consumption rate by a considerable amount.
- Multimedia-based user interaction: We are currently working on several multi-media-based (audio, visual) modules for enhanced user interaction with the system.
- Effect of noise on mining data streams: Most of the real-life data streams are noisy. Mining streams require dealing with data in an incremental manner and that makes the techniques more vulnerable to noise. We are currently exploring a collection of different approaches for dealing with noise in MobiMine.
- Estimating Fourier coefficients directly from data: This paper deals with the problem of computing the Fourier spectra of decision trees. We are currently developing techniques to accurately estimate the spectra directly from data.
- Large scale systems development and testing: We are currently exploring different systems issues for making the MobiMine capable of supporting a very large number of clients. This includes a distributed approach to process the financial data using a collection of servers.

Acknowledgments

The authors acknowledge supports from the United States National Science Foundation CAREER award IIS-0093353, NASA (NRA) NAS2-37143 and TEDCO, Maryland Technology Development Center.

References

[1] M. Ankerst, C. Elsen, M. Ester, and H. Kriegel. Visual classification: An interactive approach to decision tree construction. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 392–396, 1999.

[2] A. Azoff. *Neural Network Time Series Forecasting of Financial Markets.* Wiley, New York, 1994.

[3] N. Baba and M. Kozaki. An intelligent forecasting system of stock price using neural networks. In *Proceedings IJCNN, Baltimore, Maryland*, pages 652–657, Los Alamitos, 1992. IEEE Press.

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

[5] L. Breiman. Pasting bites together for prediction in large data sets and on-line. Technical report, 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] L. Breiman, J. H. Freidman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, CA, 1984.

[8] J. Campbell, A. Lo, and A. MacKinley. *The Econometrics of Financial Markets*. Princeton University Press, USA, 1997.

[9] S. Cheng. A neural network approach for forecasting and analyzing the price-volume relationship in the taiwan stock market. Master's thesis, National Jow-Tung University, Taiwan, R.O.C, 1994.

[10] P. Domings and G. Hulten. Mining high-speed data streams. In *Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 71–80, Boston, MA, August, 2000.

[11] H. Drucker and C. Cortes. Boosting decision trees. *Advances in Neural Information Processing Systems*, 8:479–485, 1996.

[12] W. Fan, S. Stolfo, and J. Zhang. The application of Adaboost for distributed, scalable and on-line learning. In *Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 362–366, San Diego, California, 1999.

[13] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proc. 13th International Conference on Machine Learning*, pages 148–146, Murray Hill, NJ, 1996. Morgan Kaufmann.

[14] J. Gehrke, V. Ganti, R. Ramakrishnan, and W. Loh. BOAT — optimistic decision tree construction. In *Proceedings of SIGMOD, ACM*, pages 169–180, 1999.

[15] D. Goldberg. Genetic algorithms and Walsh functions: Part I, a gentle introduction. *Complex Systems*, 3(2):129–152, 1989.

[16] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

[17] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining*, pages 97–106, San Francisco, CA, 2001. ACM Press.

[18] G. Jang, F. Lsi, and T. Parng. Intelligent stock trading decision support system using dual adaptive-structure neural networks. *Journal of Information Science Engineering*, 9:271–297, 1993.

[19] H. Kargupta, B. Park, D. Hershberger, and E. Johnson. Collective data mining: a new perspective towards distributed data mining. In *Advances in Distributed and Parallel Knowledge Discovery, Eds: Kargupta, Hillol and Chan, Philip*, pages 133–184. AAAI/MIT Press, 2000.

[20] H. Kargupta, B. Park, S. Pittie, L. Liu, D. Kushraj, and K. Sarkar. Mobimine: Monitoring the stock market from a PDA. *ACM SIGKDD Explorations*, 3:37–47, 2001.

[21] R. Kuo, L. Lee, and C. Lee. Intelligent stock market forecasting system through artificial neural networks and fuzzy delphi. In *Proceedings of World Congress on Neural Networks*, pages 345–350, San Deigo, 1996. INNS Press.

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

[23] C. Lee. Intelligent stock market forecasting system through artificial neural networks and fuzzy delphi. Master's thesis, Kaohsiung Polytechnic Institute, Taiwan, R.O.C, 1996.

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

[25] 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.

[26] D. Margineantu and T. Dietterich. Pruning adaptive boosting. In *Proceedings, Fourteenth Intl. Conf. Machine Learning*, pages 211–218, 1997.

[27] C. Merz and M. Pazzani. A principal components approach to combining regression estimates. *Machine Learning*, 36(1–2):9–32, 1999.

[28] B. Park. *Knowledge Discovery from Heterogeneous Data Streams Using Fourier Spectrum of Decision Trees.* PhD thesis, Washington State University, 2001.

[29] B. Park, R. Ayyagari, and H. Kargupta. A Fourier analysis-based approach to learn classifier from distributed heterogeneous data. In *Proceedings of the First SIAM Internation Conference on Data Mining*, Chicago, US, 2001.

[30] B. Park and H. Kargupta. Constructing simpler trees from ensemble models using Fourier analysis. In *ACM SIGMOD Workshop on Data Mining and Knowledge Discovery*, Madison, WI, June 2002.

[31] B. Park and H. Kargupta. The Fourier spectrum of decision trees: Theoretical issues and application in ensemble-based learning from data streams. Technical Report Technical Report CS-TR-02-05, Department of Computer Science, University of Maryland Baltimore-County, 2002.

[32] M. Perrone and L. Cooper. When networks disagree: Ensemble method for neural networks. In R. J. Mammone, editor, *Neural Networks for Speech and Image processing.* Chapman-Hall, 1993.

[33] A. Prodromidis, S. Stolfo, and P. Chan. Pruning classifiers in a distributed meta-learning system. In *Proceedings of the First National Conference on New Information Technologies*, pages 151–160, 1998.

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

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

[36] J. R. Quinlan. Bagging, boosting and C4.5. In *Proceedings of AAAI'96 National Conference on Artificial Intelligence*, pages 725–730, 1996.

[37] J. Schlimmer and R. Granger Jr. Beyond incremental processing: Tracking concept drift. In *AAAI, Vol. 1*, pages 502–507, 1986.

[38] W. N. Street and Y. Kim. A Streaming Ensemble Algorithm (SEA) for large-scale classificaiton. In *Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 377–382, San Francisco, CA, 2001.

[39] P. Utgoff. ID5: an incremental ID3. In J. Laird, editor, *Proceedings of the Fifth International Conference on Machine Learning*, pages 107–120, San Mateo, CA, 1988. Morgan Kaufmann.

[40] P. Utgoff. Incremental induction of decision trees. *Machine Learning*, 4:161–186, 1989.

[41] P. Utgoff. An improved algorithm for incremental induction of decision trees. In *Proc. 11th International Conference on Machine Learning*, pages 318–325. Morgan Kaufmann, 1994.

[42] J. Zirilli. *Financial Prediction Using Neural Networks.* International Thomson Computer Press, 1997.