A Fourier Spectrum-based Approach to Aggregate and Visualize Decision Trees for Mobile Applications*

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 brief overview of a novel Fourier analysis-based technique to visualize, manipulate, and aggregate decision trees. Fourier representation of a decision tree has several useful 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 presents a technique to aggregate decision trees in their Fourier representations. It also describes a touch-pad/ticker-based approach to visualize decision trees using their Fourier spectrum and an implementation for PDAs.

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[4], ID3[27], and C4.5 [28]) 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 [8, 11, 20, 29] like decision trees. From a data mining perspective it is important that these models are properly aggregated. This is because different data blocks observed at different time frames may generate different models that may actually belong to a single model which can be generated when all the data blocks are combined and mined together. We need on-line data mining algorithms that can easily aggregate and evolve models.

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,

* Patent application pending

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 Fourier basis offers an interesting representation of decision trees that can facilitate quick aggregation of a large number of decision trees [13, 24] and their visualization in a small screen using a novel "decision tree-ticker". Although we present the material in the context of devices with small display areas, the approach is also useful for desktop applications.

Section 2 explains the Fourier basis representation of a decision tree and presents an algorithm to compute the Fourier spectrum of a decision tree. Section 3 considers aggregation of multiple trees in Fourier representation. Section 4 presents a decision tree visualization technique using a touch-pad and a ticker. Section 4.3 describes an application of this technology for mining stock data streams. Finally, Section 5 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 by numeric values in a consistent manner. See Figures 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 visualizing and updating decision trees on a regular basis.

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 basis offers 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 function. Consider the function space over the set of all ℓ-bit Boolean feature vectors. The Fourier basis set that spans this space is comprised of 2^ℓ Fourier basis functions (we consider only discrete Boolean Fourier basis here). Each Fourier basis function is defined as $\psi_{\mathbf{j}}(\mathbf{x}) = (-1)^{(\mathbf{x}\cdot\mathbf{j})}$. Where $\mathbf{j}$ and $\mathbf{x}$ are binary strings of length ℓ. In other words $\mathbf{j} = (j_1, j_2, \cdots j_\ell)$, $\mathbf{x} = (x_1, x_2, \cdots x_\ell)$ and $\mathbf{j}, \mathbf{x} \in \{0,1\}^\ell$; $\mathbf{x}\cdot\mathbf{j}$ denotes the inner product of $\mathbf{x}$ and $\mathbf{j}$. $\psi_{\mathbf{j}}(\mathbf{x})$ 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 features corresponding to the 1-s in the partition $\mathbf{j}$. A function $f : \mathbf{X}^\ell \rightarrow \Re$, that maps an ℓ-dimensional space of binary strings to a real-valued range, can be written using the Fourier basis functions: $f(\mathbf{x}) = \sum_{\mathbf{j}} w_{\mathbf{j}}\psi_{\mathbf{j}}(\mathbf{x})$; where $w_{\mathbf{j}}$ is the Fourier coefficient corresponding to the partition $\mathbf{j}$; $w_{\mathbf{j}} = \frac{1}{2^\ell}\sum_{\mathbf{x}} f(\mathbf{x})\psi_{\mathbf{j}}(\mathbf{x})$. 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 then we may consider the $\mathbf{j}$-th partition to be insignificant and neglect its contribution.

2.2 Fourier Spectrum for Domains with Non-binary Features

The Fourier basis can be easily extended to the non-Boolean case. Consider a domain defined by ℓ possibly non-Boolean features where the i-th feature can take λ_i distinct values. Let $\overline{\lambda} = \lambda_1, \lambda_2 \cdots \lambda_\ell$. The generalized Fourier basis function over an $\overline{\lambda}$-ary feature space is defined as,

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

When $\lambda_1 = \lambda_2 = \cdots = \lambda_\ell = \lambda$ we can write, $\psi_{\mathbf{j}}^{(\lambda)}(\mathbf{x}) = \exp^{\frac{2\pi i}{\lambda}(\mathbf{x}\cdot\mathbf{j})}$. Where $\mathbf{j}$ and $\mathbf{x}$ are λ-ary strings of length ℓ.

The Fourier coefficients for non-Boolean domain can be defined as follows:

$$w_{\mathbf{j}} = \Pi_{i=1}^{l} \frac{1}{\lambda_i} \sum_{\mathbf{x}} f(\mathbf{x}) \overline{\psi}_{\mathbf{j}}^{(\overline{\lambda})}(\mathbf{x}) \tag{2}$$

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

Since a decision tree is a function defined over a discrete space (inherently discrete or some discretization of a continuous space) we can compute its Fourier transformation. We shall discuss the techniques to compute the Fourier spectrum

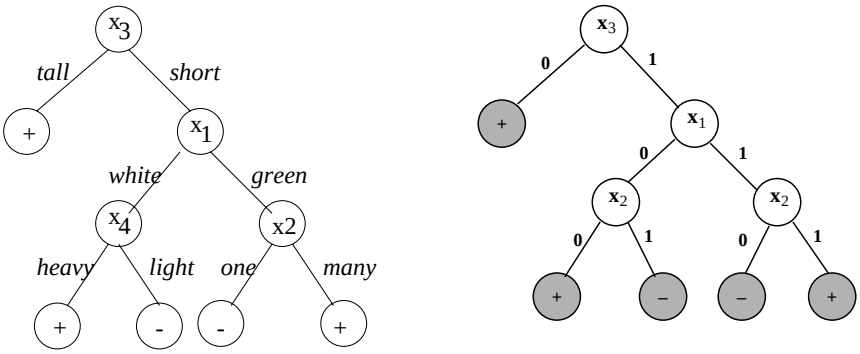

Fig. 1. (Left) A symbolic decision tree and (Right) the numeric (Boolean) version that simply replaces the symbols by numbers.

of a decision tree later. It turns out that the Fourier representation of a decision tree with bounded depth has some very interesting properties [17, 19]. 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.

1. The Fourier representation of a bounded depth (say k) Boolean decision tree only has a polynomial number of non-zero coefficients; all coefficients corresponding to partitions involving more than k feature variables are zero.
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 [19] for Boolean decision trees.

These observations suggest that the spectrum of the decision tree can be approximated by computing only a small number of low-order[1]. So Fourier basis offers an efficient numeric representation of a decision tree in terms of an algebraic function that can be easily stored and manipulated.

[1] Order of a coefficient is the number of features defining the corresponding partition. Low-order coefficients are the ones for which the orders of the partitions are relatively small

2.4 From Fourier Coefficients to a Decision Tree

The Fourier coefficients do have important physical meanings. Recall that every coefficient is associated with a Fourier basis function. Any given basis function depends on a unique subset of features defining the domain. For an ℓ-bit Boolean domain there are 2^ℓ unique feature subsets. There are also 2^ℓ different Fourier basis functions and each of them is associated with a unique subset. The magnitude of a coefficient represents the "strength" of the contribution of the corresponding subset of features to the overall function value. So if the magnitude of a coefficient is relatively large then the corresponding feature-subset have strong influence together on the function value. For example, consider a linear function of the form $f(\mathbf{x}) = \sum_i a_i x_i$. All terms in this function are linear; so the features together (in a multiplicative sense) do not make any contribution to the function value. This linearity is also reflected in its Fourier spectrum. It is easy to show that all Fourier coefficients of this function corresponding to basis functions that depend on more than one variable are zero. This connection between the structure of the function and its spectrum is a general property of the Fourier basis. The magnitudes of the Fourier coefficients expose the underlying structure of the function by identifying the dependencies and correlation among different features.

However, Fourier spectrum of a decision tree tells us more than the interactions among the features. This coefficients can also tell us about the distribution of class labels at any node of the decision tree. Recall that any node in the 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 Boolean domain, then $\mathbf{h} \in \{0, 1, *\}^\ell$, where $*$ denotes a wild-card that matches any value of the corresponding feature. For example, the path $\{(x_3 \stackrel{1}{\rightarrow} x_1, x_1 \stackrel{0}{\rightarrow} x_2\}$ 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 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 "skewedness" in the distribution for a given schema is measured by computing the *information-gain* defined in the following.

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

where $Values(x_i)$ is the set of all possible values for attribute x_i, and $|\mathbf{h}_v|$ is the set of those members of $\mathbf{h}$ that have a value v for attribute x_i; if p and q are the proportions of the positive and negative instances in some $\mathbf{h}$, then $Entropy(\mathbf{h}) = -p \log p - q \log q$.

The gain computation clearly depends on the proportion of class labels in a schema which can be determined directly from the Fourier spectrum of the tree. For example consider a problem with Boolean class labels $\{0, 1\}$. The total number of members of schema $\mathbf{h}$ with class labels 1,

$$n_1(\mathbf{h}) = \sum_{\mathbf{j} \in \mathbf{J}(\mathbf{h})} w_{\mathbf{j}} \psi_{\mathbf{j}}(\beta(\mathbf{h})) \tag{3}$$

where,

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

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

Let n_f be the total number of features that are set to specific values in order to define the schema and *ell* be the total number of features defining the entire domain. So the total number of members covered by the schema $\mathbf{h}$ is $2^{\ell - n_f}$. The total number of members in $\mathbf{h}$ with class label 0, $n_0 = 2^{\ell - n_f} - n_1$. Clearly, we can compute this distribution information for any given schema using the spectrum of the tree. In other words, we can construct the tree using the Fourier spectrum of the information gain. The following section presents a fast technique for computing the Fourier spectrum of a decision tree.

2.5 Computing the Fourier Transform of a Decision Tree

The Fourier spectrum of a decision tree can be easily computed by traversing the leaf nodes in a systematic fashion. This section presents the algorithm for doing that. The proposed algorithm is extremely fast and the overhead of computing these coefficients is minimal compared to the load for learning the tree. Let Λ be the complete instance space. Let us also assume that there are n nodes in the decision tree and the i-th leaf node covers a subset of Λ, denoted by S_{l_i}. Therefore, $\cup_{i=1}^{n} S_{l_i} = \Lambda$.

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

$$\begin{aligned} w_{\mathbf{j}} &= \frac{1}{|\Lambda|} \sum_{\mathbf{x} \in \Lambda} f(\mathbf{x}) \psi_{\mathbf{j}}(\mathbf{x}) \\ &= \frac{1}{|\Lambda|} \sum_{\mathbf{x} \in S_{l_1}} f(\mathbf{x}) \psi_{\mathbf{j}}(\mathbf{x}) + \frac{1}{|\Lambda|} \sum_{\mathbf{x} \in S_{l_2}} f(\mathbf{x}) \psi_{\mathbf{j}}(\mathbf{x}) + ... + \frac{1}{|\Lambda|} \sum_{\mathbf{x} \in S_{l_n}} f(\mathbf{x}) \psi_{\mathbf{j}}(\mathbf{x}) \end{aligned}$$

Now note that both $f(\mathbf{x})$ and $\psi_{\mathbf{j}}(\mathbf{x})$ take a constant value for every member $\mathbf{x}$ in S_{l_i}. Since the path to a leaf node represents a schema, with some abuse of symbols we can write,

$$w_{\mathbf{j}} = \frac{|S_{l_1}|}{|\Lambda|} f(\mathbf{h_1})\psi_{\mathbf{j}}(\mathbf{h_1}) + \frac{|S_{l_2}|}{|\Lambda|} f(\mathbf{h_2})\psi_{\mathbf{j}}(\mathbf{h_2}) + ... + \frac{|S_{l_n}|}{|\Lambda|} f(\mathbf{h_n})\psi_{\mathbf{j}}(\mathbf{h_n})$$

Where $\mathbf{h_i}$ is a schema defined by a path to l_i respectively. Further details about this algorithm can be found elsewhere [14]. For each path from the root to a leaf node(schema $\mathbf{h}$), all non-zero Fourier coefficient are detected by enumerating all possible value for each attribute in $\mathbf{h}$.

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. Let us define $f(\mathbf{x})$ as a classification function of an ensemble classifier. $f(\mathbf{x})$ is essentially linear weighted combination of classifications of multiple trees.

$$\begin{aligned} f(\mathbf{x}) &= a_1 f_1(\mathbf{x}) + a_2 f_2(\mathbf{x}) + ... + a_n f_n(\mathbf{x}) \\ &= a_1 \sum_{j \in J_1} w_{\mathbf{j}}^{(1)} \overline{\psi_{\mathbf{j}}}(\mathbf{x}) + a_2 \sum_{j \in J_2} w_{\mathbf{j}}^{(2)} \overline{\psi_{\mathbf{j}}}(\mathbf{x}) + ... + a_n \sum_{j \in J_n} w_{\mathbf{j}}^{(n)} \overline{\psi_{\mathbf{j}}}(\mathbf{x}) \end{aligned}$$

where $f_i(\mathbf{x})$ and a_i are i^{th} decision tree and its weight respectively. J_i is set of non-zero Fourier coefficients that are detected by i^{th} decision tree and $w_{\mathbf{j}}^{(i)}$ is a Fourier coefficient in J_i. Now we can write,

$$f(\mathbf{x}) = \sum_{j \in J} w_{\mathbf{j}} \overline{\psi_{\mathbf{j}}}(\mathbf{x}) \tag{4}$$

where $w_{\mathbf{j}} = \sum_{i=1}^{n} a_i w_{\mathbf{j}}^{(i)}$ and $J = \cup_{i=1}^{n} J_i$.

Therefore Fourier spectrum of $f(\mathbf{x})$ (an ensemble classifier) is simply the weighted spectrum of each tree. The spectrum of the ensemble can be directly useful for at least two immediate purposes:

1. visualization of the ensemble through its Fourier spectrum,
2. understanding the ensemble classifier through its spectrum, and
3. construction of simpler and smaller ensemble (possibly a single concise tree) from the aggregated spectrum.

In this paper we focus only on the first two issues.

4 Visualization of Decision Trees in Fourier Representation

Fourier basis offers a unique way to visualize decision trees. This section presents a ticker and touchpad-based approach to visualize decision trees that are especially suitable for smaller displays like the ones that we get in palmtop devices. However, the approach is also suitable for desktop applications.

4.1 Fourier Coefficient-based Touch-pad and a Ticker

Fourier coefficients of a function provide us a lot of useful information. As we noted earlier, individual coefficients tell us about strongly mutually interacting features that significantly contribute toward the overall function value. Moreover, different subsets of these coefficients can also provide us the distribution information (i.e. "information-gain" value used in ID3/C4.5) at any node. This section develops a novel approach for visualizing decision trees that exploits both of these properties of the Fourier spectrum. The approach is based on two primary components:

1. A *touch-pad* that presents a 2D density plot of all the significant coefficients and
2. a ticker that allows continuous monitoring of information-gain produced by a set of classifiers constructed by a certain group of features (possibly selected using the touch-pad).

Each of them is described in details next.

The Touch-pad The *touch-pad* is a rectangular region that presents a graphical interface to interact with the distribution of Fourier coefficients. Let us consider any arbitrary set of m coefficients. These coefficients can be viewed as a graph (V, E) where every node in V is associated with a unique coefficient; E is the set of edges where every edge between node v_i and v_j is associated with the "distance" between their associated partitions. The distance metric can be chosen in different ways. For example, if the domain is binary, partitions are also going to binary strings. So we may choose hamming distance as the metric for defining the graph. For discrete non-binary partitions we may chose any of the widely known distance metrics. The choice of distance metric does not fundamentally change the proposed approach. Any reasonable distance metric that captures the distance between a pair of integer strings (i.e. partitions) should work fine with the proposed visualization approach.

Once the graph (V, E) is defined, our next task is to project it to a two dimensional space in such a way that "neighbors" in the original high dimensional space do remain "neighbors" in the projected space. In other words, we would like to have near isometric projections where $\frac{\rho(v_i, v_j)}{\rho(v_i', v_j')} = \alpha$ for all pairs i and j and constant value of α. This work makes use of existing off-the-shelf algorithms to construct this projection. It uses the widely known self organizing feature map (SOM) [15, 16, 18] for this purpose. The SOM takes the set of m Fourier coefficients and maps them to a two dimensional grid where coefficients with similar partitions are located in neighboring positions. Although, this particular approach uses the SOM for the 2-D projection, any other technique (possibly greedy algorithms) should work fine as long as the relative distance between the nodes are reasonably preserved. The touch-pad is also color coded in order to properly convey the distribution of significant and insignificant coefficients.

Figure 2 shows a screen shot of the implementations of the touch-pad in a HP Jornada palmtop device.

The touch-pad is also interactive. Since the display area is usually small and m can be relatively large, the touch-pad offers variable degrees of resolutions. The user can interactively zoom-in or zoom-out of a specific region. The user can also find out the subset of features defining a certain part of the touch-pad by interactively choosing a region from the screen. By marking a certain region of interest, the user will get a better understanding about interesting patterns among features residing and their finer and more detailed visualization. This interactive feature of the touch-pad also gives the user an opportunity to explore the dependencies among selected subset of features with the aid of a dynamically changing "ticker". The following section describes this in detail.

4.2 The Ticker

The ticker is provided to allow continuous monitoring of specific classifiers produced by the decision trees. The ticker shows the "strengths" (measures like information-gain used frequently in the decision tree literature) for a subset of "good" decision rules produced by the trees. Figure 2 shows screen shot of the ticker depicting the strengths of a set of classifiers at a particular time. As new data arrive the strengths are modified.

It is inherently designed to complement the touch-pad-based approach to visualize the spectrum of the decision tree. Recall that the information-gain (i.e. difference in the entropies) at a particular node of a decision tree is computed by using the distribution of the domain members that it covers. As we noted earlier, the information-gain can also be computed from the spectrum of the decision tree. User's selection of a certain region from the touch-pad identifies the current interest of the user to a certain subset of features. The user can invoke the ticker and instruct it to monitor all the decision rules that can be constructed using those features. The user can also interactively select the decision rules that are comprised of different features. The ticker in turn collects the relevant Fourier coefficients and computes the information-gain associated with all the decision rules that can be constructed using those features and shows only the good ones.

4.3 Aggregating Multiple Trees Generated from Financial Data Streams

This section presents the experimental performance of the proposed aggregation approach for combining multiple decision trees. It presents results using two ensemble learning techniques, namely Bagging and Arcing.

The ensemble learning literature considers different ways to compute the output of the ensemble. Averaging the outputs of the individual models with uniform weight is probably the simplest possibility. Perrone and Cooper [25][22] refer to this method as Basic Ensemble Method (BEM) or naive Bagging. Breiman proposed an Arcing method Arc-fx [5][3] for mining from large data set and stream data . It is fundamentally based on the idea of Arcing – adaptive re-sampling

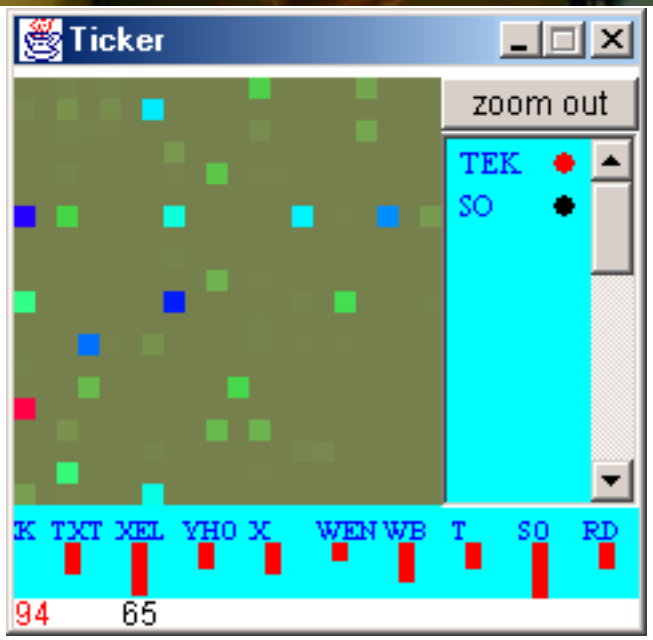

Fig. 2. (Top) Fourier spectrum-based decision tree mining interface for HP Jornada. (Bottom) An enlarged view of the interface. The interface has three windows. The leftmost window is the touch-pad. The bottom window shows the ticker. The right window is a small area to interactively construct and compute the accuracy of the classifiers.

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 [9]. 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 use.

Not all the models generated from different data blocks should always be aggregated together. Sometimes we may want to use a pruning algorithm [21, 26] for selecting the right subset of models. Sometimes a "windowing scheme" [9, 3] can be used where only W most recent classifiers are used for learning and classification.

Our experiments were performed using a semi-synthetic data stream with 174 Boolean attributes. The objective is to continuously evolve a decision tree-based predictive 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 are 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 predict whether the Yahoo stock is likely to increase or decrease 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 add white noise to the generator. Test data set is comprised of $10,000$ instances.

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 spectrums 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 3 (a) 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, while Arcing shows gradual increase throughout all iteration steps. Although we stopped at 300 iterations, the accuracy does not seem to reach its asymptotic value. Figure 3 (b) 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. Figure 3 (c) plots the classification accuracies of Bagging and Arcing

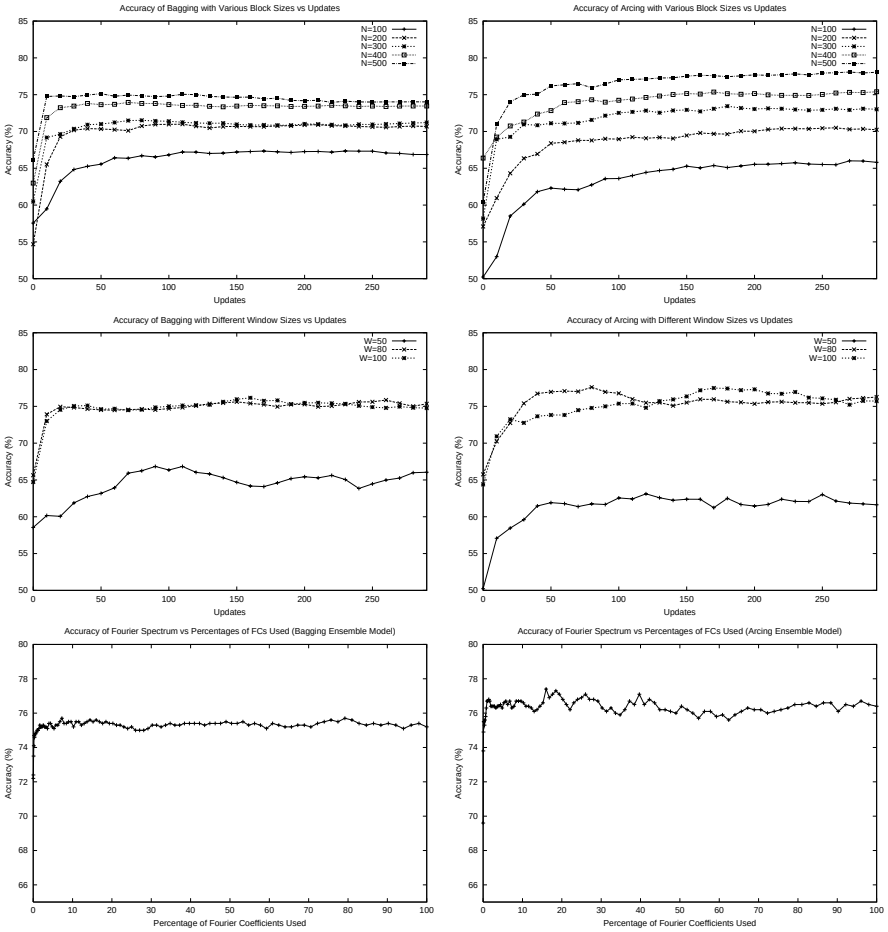

Fig. 3. (a) Accuracy vs. iterations with various data block sizes: (Left) Bagging (Right) Arcing. (b) Accuracy vs. iterations with various window sizes: (Left) Bagging (Right) Arcing. (c) Accuracy vs. percentage of Fourier coefficients used for prediction: (Left) Bagging, (Right) Arcing.

using different significant fractions of the Fourier spectrum. It shows that only a small fraction of all the coefficients in the spectrum is sufficient for accurate classification. Further details about these and additional experiments studying the complexity of the Fourier representations of decision trees can be found elsewhere [14].

5 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 expectations 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 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. The approach presented here is particularly suitable for portable applications with small display areas. The touch-pad and ticker-based approach is very intuitive and can be easily implemented on touch sensitive screen often used in small wireless devices. We are currently extending the ticker interface, introducing different additional application functionalities, and exploring related techniques to mine real-valued financial data online.

Acknowledgments

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

References

1. M. Ankerst, C. Elsen, M. Ester, and H. Kriegel. Visual classication: An interactive approach to decision tree construction. *Proceeding of the 5th International Conference on Knowledge Discovery and Data Mining*, pages 392–397, 1999.
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. Pasting bites together for prediction in large data sets and on-line. Technical report, Statistics Department, University of California at Berkeley, 1997.

4. L. Breiman, J. Freidman, R. Olshen, and C. Stone. Classification and regression trees. 1984.
5. L. Breiman. Bias, variance and arcing classifiers. Technical Report 460, Statistics Department, University of California at Berkeley, 1996.
6. L. Breiman. Prediction games and arcing classifiers. Technical Report 504, Statistics Department, University of California at Berkeley, 1997.
7. P. Domings and G. Hulten. Mining high-speed data streams. In *Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Boston, MA, August, 2000.
8. H. Drucker and C. Cortes. Boosting decision trees. *Advances in Neural Information Processing Systems*, 8:479–485, 1996.
9. 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*, San Diego, California, 1999.
10. Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.
11. Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, Murray Hill, NJ, 1996.
12. Y. Freund and R. Schapire. A decision theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):97–119, 1997.
13. H. Kargupta, B. Park, D. Hershberger, and E. Johnson. Collective data mining: a new perspective towards distributed data mining Advances in Distributed and Parallel Knowledge Discovery. Editors: H. Kargupta and P. Chan, pages 133–184, AAAI/MIT Press.
14. H. Kargupta and B. Park. Mining time-critical data streams using the Fourier spectrum of decision trees. (In communication), 2001.
15. S. Kaski. Fast winner search for SOM-based monitoring and retrieval of high-dimensional data. In *Proceedings of ICANN99, Ninth International Conference on Artificial Neural Networks*, volume 2, pages 940–945. IEE, London, 1999.
16. T. Kohonen. The self–organizing map. *Proceeding of the IEEE*, 78(9):1464–1479, 1990.
17. 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.
18. K. Lagus, T. Honkela, S. Kaski, and T. Kohonen. WEBSOM for textual data mining. *Artificial Intelligence Review*, 13(5/6):345–364, December 1999.
19. N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, Fourier transform, and learnability. *Journal of the ACM*, 40:607–620, 1993.
20. 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.
21. D. Margineantu and T. Dietterich. Pruning adaptive boosting. In *Proceedings, Fourteenth Intl. Conf. Machine Learning*, pages 211–218, 1997.
22. C. Merz and M. Pazzani. A principal components approach to combining regression estimates. *Machine Learning*, 36(1–2):9–32, 1999.
23. D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.
24. B. Park, R. Ayyagari, and H. Kargupta. Proceedings of the SIAM International Conference on Data Mining.

25. 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.
26. A. Prodromidis and S. Stolfo. Mining databases with different schemas: Integrating incompatible classifiers. In R. et al. Agrawal, editor, *The Fourth International Conference on Knowledge Discovery and Data Mining*, pages 314–318. AAAI Press, 1998.
27. J. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.
28. J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kauffman, 1993.
29. J. Quinlan. Bagging, boosting and C4.5. In *Proceedings of AAAI'96 National Conference on Artificial Intelligence*, pages 725–730, 1996.
30. R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, pages 297–336, 1999.
31. P. Utgoff. Incremental induction of decision trees. *Machine Learning*, 4:161–186, 1989.
32. D. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.