Constructing Simpler Decision Trees from the Fourier Spectrum of Ensemble Models: Theoretical Issues and Application in Mining Data Streams

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

Abstract

Ensemble learning approach is a natural candidate to mine data streams. It handles the incrementally observed data by generating a stream of models constructed from different sets of observations. These models together capture the behavior of the stream. However, the complexity of the ensemble models for data streams tends to increase linearly with time as new models are continuously added. Moreover, large ensembles are often hard to interpret and difficult to translate into action-able knowledge. This paper offers a novel Fourier spectrum-based technique to construct a possibly simpler tree from a decision tree-ensemble generated from data streams. It discusses several theoretical properties of the Fourier spectrum of decision tree ensembles and extends the proposed approach to deal with domains with more than two class labels. It also presents extensive experimental results to support the theoretical claims.

1 Introduction

Everyday, large volumes of data are generated from constantly operating sources. Faced with this continuous flow of data, often called *data streams*, the challenge is how to process the huge volume of data in a timely and efficient manner. For example, the NASA Earth Observing System (EOS) generates more than 100 gigabytes of image data per hour that is stored, managed, and distributed by the EOS Data and Information System (EOSDIS). The Internet also provides various sources of stream data. Many important financial data are regularly posted on the Internet, including announcements regarding new products, expected earnings, legal battles, and merges. In addition, sites such as NASDAQ, Yahoo finance, and CNN finance offer near-realtime stock data including up-to-date company financial profiles. "Click stream" data from the web sites, telephone communication data, sensor networks data are additional examples of the data stream scenario considered here.

Mining time-critical data streams usually requires on-line or ensemble-based algorithms that often produce a series of models [7, 9, 19, 26] like decision trees. In both cases, data is often collected in blocks. The ensemble approach can quickly generate a partial model based on the recent block of data. Since such slices of data stream only contain limited (or, incorrect) knowledge, the ensemble approach heavily relies on aggregating outputs of partial models. Naturally, the model complexity of an ensemble increases linearly as partial models are added. Therefore, it is crucial to maintain the ensemble in its simplest form.

Our earlier work [14, 15] reported the use of a novel Fourier analysis based approach to aggregate multiple decision trees in the Fourier representation. That work showed that a decision tree can be treated as a function and therefore we can compute its Fourier spectrum. It is also important to note that the spectrum can be computed in a computationally efficient manner. Therefore, the trees generated by the ensemble learner can be quickly transformed into their respective Fourier spectrum and then the spectra can be aggregated (e.g. average, weighted average) in the Fourier domain. The aggregated model in the Fourier domain can be used as a classifier, which is functionally equivalent to the ensemble model. The work also showed that the aggregated Fourier spectrum can also be used to visualize decision tree ensembles and to identify emerging patterns by noting the significant Fourier coefficients.

This paper further explores the work and contributes important results. It offers theoretical results for the Fourier spectrum of non-Boolean decision trees. It extends the approach to decision trees with multiple class labels. It also offers a novel way to construct decision trees from the Fourier spectrum of the ensemble model. Although the spectrum-based representation of an ensemble is functionally equivalent to the ensemble model, computing the output of the classifier requires a considerable floating point operations. Moreover, in some applications constructing an equivalent decision tree from the tree-ensemble is more natural. The proposed technique to construct a tree from the ensemble-spectrum allow us to solve both the problems. So now we can first compute the spectrum of the individual trees of the ensemble, aggregate them in the Fourier representation, and then go back to the tree domain by constructing the tree using the Fourier coefficients. This is very analogous to going back and forth between time and frequency domains of the traditional Fourier analysis in signal processing domain.

The rest of the paper is organized as follows. Section 2 reviews the algorithms that mine from data stream environments. Section 3 introduces the Fourier transform of a decision tree. It discusses several theoretical properties of the Fourier spectrum of decision trees. Section 4 describes an algorithm to compute Fourier spectrum of decision tree. Section 5 presents the algorithm for constructing a decision tree from the ensemble spectrum. Section 6 offers empirical verification of the analytical claims. Finally, Section 7 concludes the paper.

2 Mining Data Streams: Prior Related Works

Mining Data streams offer several challenges. It requires mining algorithms that can handle incrementally observed data. Both incremental and ensemble learning-based techniques are natural candidates to mine data streams. While the former aims to adapt the the existing model structure based on new observations, the latter generates a stream of partial models for different sets of observations and then combines the partial models into a single ensemble model. This section reviews the related literature from each of these two categories for mining data streams.

2.1 Incremental Learning

Incremental techniques have been developed for both unsupervised and supervised learning. Although incremental unsupervised clustering techniques are very important, in this section we primarily consider supervised classifier learning techniques since that is the focus of this paper.

Most of the incremental classifier learning algorithms work in the following manner. First a classifier is built from the already observed data set. This model classifies the new observations and it immediately receives the true classifications from an existing oracle. Any discrepancy between the outputs of the current model and the true outputs are fed into the learning algorithm; it then produces a modified classifier that tries to minimize the differences.

There exists a large number of incremental classifier learning algorithms. Schlimmer and Granger [27] pointed out the need for such algorithms by noting that changes in the hidden context can produce the *concept drift*, which in turn has a dramatic impact on the learning process. Decision trees [25] are widely used for classification. Conventional decision tree learning systems are *batch* algorithms. They require all the training instances be available in the from of static data set. In order to deal with the data stream scenario, several incremental decision tree learning algorithms have been developed. They include ID4[27], ID5[29], ID5R[30] and ITI[31]. Like ID3 [25], all these systems use information gain [25] as their measure to select attributes for decison nodes. They also use chi-square tests to prune the tree. They are all designed to incrementally build a decision tree by accepting one training instance at a time. In order to do this, they keep necessary statistics (measure for information gain) at each decision node. IDR5, in particular, is designed to build the same decision tree that ID3 would construct. An improved system, ITI [31], builds a decision tree in a similar fashion; however, it can handle both nominal and continuous-valued attributes. IDL [4] offers another variety of incremental decision tree learner. It is similar to ID5; however, it adopts a topological criterion to minimize the number of occurrences of each attribute along each path.

Recently, BOAT [10], VFDT [6] and its successor CVFDT [13] have been introduced as incremental decision tree induction algorithms. The Bootstrapped Optimistic Algorithm for Tree Construction (BOAT) is a two-stage decision tree building algorithm that is based on bootstrap sampling. During the first stage it constructs a decision tree using only a small subset of the data. All splitting criteria that are obtained during the first stage are expected to be close to the optima. Any incorrect splitting criterion is corrected during the second stage, which is called the "cleanup stage." Like IDR5, BOAT maintains a decision tree equivalent to what would be constructed with the presence of the entire data set. However, since it tries to build the exact tree (learn-able from the entire data), it discards and regrows some portions of the tree when it observes a drift in the underlying distribution. Therefore, it performs slowly when the drift is large.

VFDT [6] is designed to learn from large quantity of data efficiently without requiring large in-core memory. It particularly notes that only a small sample may be sufficient to choose the split attribute at any given node. To determine a proper sample size, it uses a statistical result known as the *Hoeffding* or *additive Chernoff bound* [12]. However, VFDT cannot deal with concept drift, i.e. non-stationary distributions. In other words, it assumes that the data is a random sample from a stationary distribution. Hulten et al. [13] proposed CVFDT, an extension to VFDT, to overcome this weakness of VFDT. In order to detect and respond to changes in the data stream, CVFDT keeps its model consistent with a sliding window of examples. The window slides as new instances are observed; thus, it increments the counts corresponding to the recent observations and decrements the counts corresponding to the older ones. Instead of constructing a new model as the window slides, it grows an alternative subtree when some previous splits no longer achieve higher gain. An alternative subtree replaces the old one when it becomes more accurate. The following section reviews the existing ensemble-based approaches for classifier learning from data streams.

2.2 Ensemble Classifier Model

Ensemble-based classifiers work by generating a collection of base classifiers and combining their outputs using some pre-sprecified schemes. Typically, voting (weighted or unweighted) schemes are employed to combine the outputs of the base classifiers. A large volume of research reports that ensemble classifier models often perform better than any of the base classifiers used [5, 22, 1, 21, 17].

In data stream environments, an ensemble-based approach would generate a stream of classifiers from incrementally collected different data blocks. It would generate some classifier C_k from the data block D_k, observed during a given time period t_k. The challenge is to combine C_k with previously learned classifiers $C_1, C_2, ..., C_{k-1}$. The simplest known approach is to combine classifiers with uniform weight (or unweighted average). Perrone and Cooper [23] refer to this method as the **B**asic **E**nsemble **M**ethod (BEM).

Several researchers have studied aggregation of classifiers with non-uniform weights. They have one thing in common — previously learned classifiers together are used to assign weights to instances in the current data block, thus giving non-uniform probabilities to be sampled in the next round.

Wei Fan and et al. proposed [8] a scheme using AdaBoost as a means to assign instance weight and adjust weights of previously learned classifiers based on their accuracies over new data block. They observed that AdaBoost provides a way to adjust weights of classifiers based on the test over the validation set.

Breiman proposed an Arcing method [2, 3] for learning from large data sets and streams. It is still based on the idea of adaptive re-sampling by giving higher weights to those instances that are often wrongly classified. This method includes a newly observed data instance into the next training data set if it is mis-classified by the previous classifiers. If not, it is accepted with the following probability,

$$\frac{e(k)}{1-e(k)}$$

where $e(k)$ is an estimate of the error of the classifiers $C_1, C_2, ..., C_k$ over the k^{th} training data set D_k. $e(k)$ is a smoothed version of $r(k)$, which is the proportion of mis-classified data instances in building k-th training data set. Here $e(k)$ is computed as follows:

$$e(k) = pe(k-1) + (1-p)r(k), \quad p \in (0,1)$$

A major drawback of models described in this section is storage overhead. In contrast with incremental learning, the size of an ensemble model grows in a linear fashion as new sets of data flow in. This gradually increases memory usage and slows down both learning and classifying processes. We may use some post-pruning algorithms for the ensemble model found in [20, 24]. However, a "windowing scheme" is reported to be preferable [8, 3]. In this scheme, only the N most recent classifiers are stored and used for both learning and classification.

Recently Street and Kim [28] proposed the Streaming Ensemble Algorihtm (SEA) that learns an ensemble of decision trees for large-scale classification. The 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.

This section reviewed incremental and ensemble classifier learning techniques to mine from data streams. The rest of the paper will focus on decision tree-based approaches to mine data streams using ensembles. As noted earlier, the ensemble-based approach would generate a stream of decision trees that are difficult to simplify and comprehend which is very important for data mining applications. The following section discusses a Fourier analysis-based approach that allows representing a large collection of decision trees in a simple and compact form. It also permits constructing a simpler decision tree from the aggregated representation that is functionally equivalent to the given ensemble.

3 Decision Trees and the Fourier Domain

This section reviews the Fourier representation of decision tree ensembles, introduced elsewhere [14, 15]. It also presents some new analytical results.

3.1 Decision Trees as Numeric Functions

A decision tree defined over a domain of categorical attributes can be treated as numeric function. First note that a decision tree is a function that maps its 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.

Once the tree is converted to a discrete numeric function, we can also apply any appropriate analytical transformation that we want. Fourier transformation is one such interesting possibility. Fourier basis offers an additive 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, called Fourier coefficients, completely define the representation. Each coefficient is associated with a Fourier basis function that depends on a certain subset of features defines the domain of the data set to be mined. The following section presents a brief review of Fourier representation.

3.2 A Brief Review of the Fourier Basis in the Boolean Domain

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. 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; $x_m, j_m \in \{0, 1, \cdots \lambda_i\}$ and $\overline{\lambda}$ represents the feature-cardinality vector, $\lambda_0, \cdots \lambda_\ell$; $\psi_{\mathbf{j}}^{\overline{\lambda}}(\mathbf{x})$ is called 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 it contains. A Fourier basis function depends on some x_i only when the corresponding $j_i \neq 0$. 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 $f : \mathbf{X}^\ell \to \Re$, 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. Energy of a spectrum is defined by the summation $\sum_{\mathbf{j}} w_{\mathbf{j}}^2$. The following section discusses some useful properties of the distribution of the enregy in the Fourier spectrum of a decision tree.

3.3 Properties of Decision Trees in the Fourier Domain

For almost all practical purposes decision trees have bounded depths. This section will therefore consider decision trees of finite depth bounded by some constant. The underlying functions in such decision trees are computable by a constant depth Boolean AND and OR circuit (or equivalently AC^0 circuit). Linial et al. [18] noted that the Fourier spectrum of AC^0 circuit has very interesting properties and proved the following lemma.

Lemma 1 (Linial, 1993) *Let M and d be the size and depth of an AC^0 circuit. Then*

$$\sum_{\{\mathbf{j} \mid o(\mathbf{j}) > t\}} w_{\mathbf{j}}^2 \leq 2M 2^{-t^{1/d}/20}$$

where $o(\mathbf{j})$ denotes the order (the number of non-zero variable) of partition $\mathbf{j}$ and t is a non-negative integer. The term on the left hand side of the inequality represents the energy of the spectrum captured by the coefficients with order greater than a given constant t.

The lemma essentially states the following properties about decision trees:

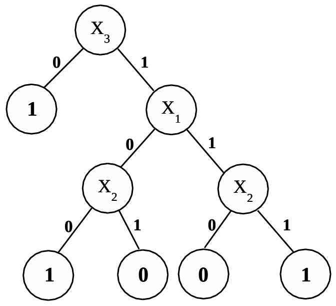

Figure 1: A Boolean decision tree

1. High order Fourier coefficients are small in magnitude.
2. The energy preserved in all high order Fourier coefficients is also small.

The key aspect of these properties is that the energy of the Fourier coefficients of higher order decays exponentially. This observation suggests that the spectrum of a Boolean decision tree (or equivalently bounded depth function) can be approximated by computing only a small number of low order Fourier coefficients. 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.

The exponential decay property of Fourier spectrum also holds for non-Boolean decision trees. The complete proof is given in the appendix.

4 Computing the Fourier Transform of a Decision Tree

The Fourier spectrum of a given tree can be computed efficiently by efficiently traversing the tree. This section first reviews an algorithm to do that. It discusses aggregation of the multiple spectra computed from the base classifiers of an ensemble. It also extends the technique for dealing with non-Boolean class labels. Kushilevitz and Mansour [16] considered the issue of learning the low order Fourier spectrum of the target function (represented by a Boolean decision tree) from a data set with uniformly distributed observations. Note that the current contribution is fundamentally different from their goal. This paper does not try to learn the spectrum directly from the data. Rather it considers the problem of computing the spectrum from the decision tree generated from the data.

4.1 Schema Representation of a Decision Path

For the sake of simplicity, let us consider a Boolean decision tree , as shown in Figure 1. The Boolean class labels correspond to positive and negative instances of the concept class. We can express a Boolean decision tree as a function $f \ : X^{\ell} \rightarrow \ \{0, 1\}$. The function f maps positive and negative instances to one and zero respectively. A node in a tree is labeled with a feature x_i. A downward link from the node x_i is labeled with an attribute value of the i-th feature. The path from the root node to a successor node represents the subset of data that satisfies the different feature values labeled along the path. These subsets of the domain are essentially similarity-based equivalence classes and we shall call them *schemata* (schema in singular form). If $\mathbf{h}$ is a schema, then $\mathbf{h} \in \{0, 1, *\}^{\ell}$, where $*$ denotes a wildcard that matches any value of the corresponding feature. For example, the path $\{(x_3 \xrightarrow{1} x_1, x_1 \xrightarrow{0} x_2\}$ in Figure 1 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. We often use the term *order* to represent the number of non-wildcard values in a schema. The following section views a decision tree in the light of schemata and describes a algorithm to extract Fourier coefficients from a tree.

4.2 Extracting and Calculating Significant Fourier Coefficients from a Tree

Considering a decision tree as a function, the Fourier transform of a decision tree can be defined as:

$$\begin{aligned} w_{\mathbf{j}} &= \frac{1}{|\Lambda|} \sum_{\mathbf{x} \in \Lambda} f(\mathbf{x}) \psi_{\mathbf{j}}(\mathbf{x}) && (1) \\ &= \frac{1}{|\Lambda|} \sum_{\mathbf{x} \in S_{l_1}} f(\mathbf{x}) \psi_{\mathbf{j}}(\mathbf{x}) + \frac{1}{|\Lambda|} \sum_{\mathbf{x} \in S_{l_2}} f(\mathbf{x}) \psi_{\mathbf{j}}(\mathbf{x}) + \ldots + \frac{1}{|\Lambda|} \sum_{\mathbf{x} \in S_{l_n}} f(\mathbf{x}) \psi_{\mathbf{j}}(\mathbf{x}) && (2) \\ &= \frac{|S_{l_1}|}{|\Lambda|} f(\mathbf{h_1}) \psi_{\mathbf{j}}(\mathbf{h_1}) + \frac{|S_{l_2}|}{|\Lambda|} f(\mathbf{h_2}) * \psi_{\mathbf{j}}(\mathbf{h_2}) + \ldots + \frac{|S_{l_n}|}{|\Lambda|} f(\mathbf{h_n}) \psi_{\mathbf{j}}(\mathbf{h_n}) && (3) \\ & && (4) \end{aligned}$$

Where Λ denotes the complete instance space, S_{l_i} is an instance subspace which i^{th} leaf node l_i covers and $\mathbf{h_i}$ is a schema defined by a path to l_i respectively (Note that any path to a node in a decision tree is essentially a subspace or hyperplane, thus it is a schema).

Lemma 2 *For any Fourier basis function $\psi_{\mathbf{j}}$, $\sum_{\mathbf{x} \in \Lambda} \psi_{\mathbf{j}}(\mathbf{x}) = 0$.*

Proof: *Since Fourier basis functions form an orthogonal set,*

$$\textstyle\sum_{\mathbf{x} \in \Lambda} \psi_{\mathbf{j}}(\mathbf{x}) = \sum_{\mathbf{x} \in \Lambda} \psi_0(\mathbf{x}) \psi_{\mathbf{j}}(\mathbf{x}) = 0.$$

Here, ψ_0 is the zero-th Fourier basis function, which is constant (one) for all $\mathbf{x}$. ■

Lemma 3 *Let $\mathbf{h_i}$ be a schema defined by the path to a leaf node l_i. Then if $\mathbf{j}$ has a non-zero attribute value at a position where $\mathbf{h_i}$ has no value (wild-card),*

$$\textstyle\sum_{\mathbf{x} \in S_{l_i}} f(\mathbf{x}) \psi_{\mathbf{j}}(\mathbf{x}) = f(\mathbf{h}_i) \sum_{\mathbf{x} \in S_{l_i}} \psi_{\mathbf{j}}(\mathbf{x}) = 0.$$

Where S_{l_i} is the subset that $\mathbf{h_i}$ covers.

Proof: *Let $\mathbf{j} = (\mathbf{j}_{in} \mathbf{j}_{out})$, where $\mathbf{j}_{in}$ are features which are included h_i and $\mathbf{j}_{out}$ are features not in h_i respectively. Since all values for $\mathbf{j}_{in}$ are fixed in $\mathbf{h_i}$, $\psi_{\mathbf{j}_{in}}(x)$ is constant for all $x \in S_{l_i}$.*

And S_{l_i} forms redundant (multiples of) complete domain with respect to $\mathbf{j}_{out}$. Therefore for a leaf node l_i,

$$\begin{aligned}
\sum_{\mathbf{x}\in S_{l_i}} f(\mathbf{x})\psi_{\mathbf{j}}(\mathbf{x}) &= \sum_{\mathbf{x}\in S_{l_i}} f(\mathbf{h_i})\psi_{\mathbf{j}}(\mathbf{x}) \\
&= f(\mathbf{h_i}) \sum_{\mathbf{x}\in S_{l_i}} \psi_{\mathbf{j}_{in}}(\mathbf{x})\psi_{\mathbf{j}_{out}}(\mathbf{x}) \\
&= f(\mathbf{h_i})\psi_{\mathbf{j}}(\mathbf{h_i}) \sum_{\mathbf{x}\in S_{l_i}} \psi_{\mathbf{j}_{out}}(\mathbf{x}) \\
&= 0 \ \blacksquare
\end{aligned}$$

Lemma 4 *For any Fourier coefficient $w_{\mathbf{j}}$ whose order is greater than the depth of a leaf node l_i, $\sum_{\mathbf{x}\in S_{l_i}} \psi_{\mathbf{j}}(\mathbf{x}) = 0$. If the order of $w_{\mathbf{j}}$ is greater than the depth of tree, then $w_{\mathbf{j}} = 0$.*

Proof: *The proof immediately follows from Lemma 3.* $\blacksquare$

Thus, for a FC $w_{\mathbf{j}}$ to be non-zero, there should exist at least one schema $\mathbf{h}$ that has non-wild-card attributes for all non-zero attributes of $\mathbf{j}$. In other words, there exists a set of non-zero FCs associated with a schema $\mathbf{h}$. This observation leads us to a direct way of detecting and calculating all non-zero FCs of a decision tree: For each schema $\mathbf{h}$ (or path) from the root, we can easily detect all non-zero FCs by enumerating all FCs associated with $\mathbf{h}$.

Before describing details of the algorithm, let us define some notations. Operator $\uplus$ is defined over a string $\mathbf{h}$ and an attribute-value pair, (x_k, i). If x_k is the k-th attribute (or feature), it outputs a new string $\mathbf{h}_k$ by replacing k-th value of $\mathbf{h}$ with i. For example, for $\mathbf{h}$ = 1** and the feature x_1, $\mathbf{h} \uplus (1,1) = 11*$. Here, we assume *indexing* starts from zero. $\uplus$ operates on both schemata and partitions.

Next, let us define a function $\delta(x_k)$, which takes in x_k as an input and outputs a set of (k, i) pairs. Here, k denotes that x_k is the k-th attributes and i is a non-zero value of x_k. If x_k has the cardinality of λ_k, then $\delta(x_k) = \{(k,1),(k,2),...,(k,\lambda_k - 1)\}$.

Let us define another operator $\otimes$ over a set of partition $\mathbf{S}$ and $\delta(x_k)$. It outputs a new set of partitions by applying $\uplus$ over all possible pairs between $\mathbf{S}$ and $\delta(x_k)$. This can be considered as Cartesian product of $\mathbf{S}$ and $\delta(x_k)$. For example, let $\mathbf{S}$ = {000,010} and x_2 be the 2nd attribute of cardinality two (with possible non-zero value of 1). Then, $\delta(x_2) = \{(2,1)\}$ and $\mathbf{S} \otimes \delta(x_2) = \{000 \uplus (2,1), 010 \uplus (2,1)\} = \{001, 011\}$.

Finally, let us consider a non-leaf node n that has d children. In other words, there exist d disjoint subtrees below n. If x_k is the feature appearing in n, then $F_{x_k}(i)$ denotes the average output value of domain members covered by a subtree accessible through the i-th child of n. For example, in Figure 2, $F_{x_1}(0)$ is $\frac{1}{2}$ and $F_{x_2}(1)$ is one. Note that $F_{x_k}(i)$ is equivalent to the average of schema $\mathbf{h}$, where $\mathbf{h}$ denotes the path (from the root node) to i-th subtree of the node where x_k appears.

The algorithm starts with pre-calculating all $F_{x_k}(i)$-s (This is essentially recursive "Tree-Visit" operation). Initially, $\mathbf{S} = \{000...0\}$ and corresponding $w_{000...0}$ is calculated with overall

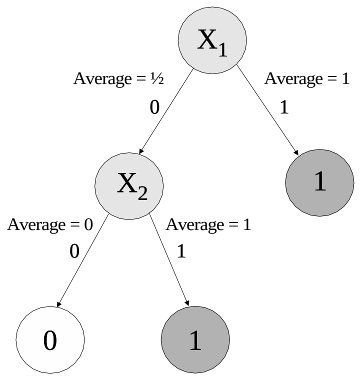

Figure 2: An instance of Boolean decision tree that shows average output values at each subtree.

average of output. In Figure 2, it is:

$$\tfrac{1}{2} \times \tfrac{1}{4} + \tfrac{1}{2} \times 1 = \tfrac{3}{4}.$$

The algorithm continues to extract all remaining non-zero FCs in recursive fashion from the root. If we assume that the tree in Figure 2 is built from data with three attributes x_0, x_1 and x_2 then we can write $\mathbf{S} \otimes \delta(x_1) = \{010\}$ and w_{010} is computed using Equation 1:

$$\begin{aligned} w_{010} &= \frac{1}{2} \times f(*0*)\psi_{010}(*0*) + \frac{1}{2} \times f(*1*)\psi_{010}(*1*) \\ &= \frac{1}{2} \times F_{x_1}(0)\psi_{010}(*0*) + \frac{1}{2} \times F_{x_1}(1)\psi_{010}(*1*) \\ &= \frac{1}{2} \times \frac{1}{2} \times 1 + \frac{1}{2} \times 1 \times (-1) \\ &= \frac{1}{4} - \frac{1}{2} = -\frac{1}{4} \end{aligned}$$

For x_2, $\mathbf{S} = \{000, 010\}$ and $\mathbf{S} \otimes \delta(x_2) = \{001, 011\}$. w_{001} and w_{011} are computed similarly as w_{010}. The pseudo code of the algorithm is presented in Figure 3.

4.3 Fourier Spectrum of an Ensemble Classifier

The Fourier spectrum of an ensemble classifier that consists of multiple decision trees can be computed by aggregating the spectra of the individual base models. Let $f(\mathbf{x})$ be the underlying function computed by a tree-ensemble where the output of the ensemble is a weighted linear combination of the outputs of the base tree-classifiers.

$$f(\mathbf{x}) = a_1 f_1(\mathbf{x}) + a_2 f_2(\mathbf{x}) + ... + a_n f_n(\mathbf{x})$$

```
1    Function ExtractFS(input:  Partition Set $\mathbf{S}$, Node $node$, Schema $\mathbf{h}$)
2      $x_k \leftarrow$ feature appearing in $node$
3      $\mathbf{N} \leftarrow \; \mathbf{S} \otimes \delta(x_k)$
4      $k \leftarrow$ position of $x_k$
5      $size \leftarrow \frac{|node|}{\lambda_k |\Lambda|}$
6      for each $\mathbf{j} \in \mathbf{N}$
7        for each possible value $i$ of $attr$
8          $\mathbf{h_i} \leftarrow \mathbf{h} \uplus (k, i)$
9          $w_{\mathbf{j}} \leftarrow w_{\mathbf{j}} + size \times F_{x_k}(i)\psi_{\mathbf{j}}(\mathbf{h}_i)$
10        end
12     end
13     $\mathbf{S} \leftarrow \mathbf{S} \cup \mathbf{N}$
14     for $i^{th}$ child node $node_i$ of $node$
15       $\mathbf{h_i} \leftarrow \mathbf{h} \uplus (k, i)$
16       ExtractFS($\mathbf{S}$,$node_i$, $\mathbf{h_i}$)
17     end
18   end
```

Figure 3: Algorithm for obtaining Fourier spectrum of a decision tree. λ_k denotes the cardinality of attribute x_k and $|node|$ denotes the size of subspace *node* covers. $|\Lambda|$ is the size of the complete instance space. k in line 4 denotes that x_k is the k-th attribute.

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

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 equation 5 is written as:

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

where $w_{\mathbf{j}} = \sum_{i=1}^{n} a_i w_{\mathbf{j}}^{(i)}$ and $J = \cup_{i=1}^{n} J_i$. The following section extends the Fourier spectrum-based approach to represent and aggregate decision trees to domains with multiple class labels.

4.4 Fourier Spectrum of Multi-Class Decision Trees

A multi-class decision tree has $k > 2$ different class labels. In general, we can assume that each label is again assigned a unique integer value. Since such decision trees are also functions that map an instance vector to numeric value, the Fourier representation of such tree is essentially not any different. However, the Fourier spectrum cannot be directly applied to represent an ensemble of decision trees that uses voting as its aggregation scheme. The Fourier spectrum faithfully represents functions in closed forms and ensemble classifiers are not such functions. Therefore, we need a different approach to model a multi-class decision trees with the Fourier basis.

Let us consider a decision tree that has k classifications. Then let us define $\Im_i$ to be the Fourier spectrum of a decision tree whose class labels are all set to zero except the i-th class. In other words, we treat the tree to have a Boolean classification with respect to the i-th class label. If we define $f^{(k)}(\mathbf{x})$ to be a partial function that computes the inverse Fourier transform using $\Im_k$, classification of an input vector $\mathbf{x}$ is written as:

$$f(\mathbf{x}) = c_1 f^{(1)}(\mathbf{x}) + c_2 f^{(2)}(\mathbf{x}) + \cdots + c_l f^{(l)}(\mathbf{x}) \tag{5}$$

where each c_i corresponds to a mapped value for the i-th classification. Note that if $\mathbf{x}$ belongs to j-th class, $f^{(i)}(\mathbf{x}) = 1$ when $i = j$, and 0 otherwise.

Now let us consider an ensemble of decision trees in weighted linear combination form. Then $f^{(k)}(\mathbf{x})$ can be written as:

$$f^{(k)}(\mathbf{x}) = a_1 f_1^{(1)}(\mathbf{x}) + a_2 f_2^{(2)}(\mathbf{x}) + \cdots a_l f_l^{(l)}(\mathbf{x})$$

where a_i and $f_i^{(k)}(\mathbf{x})$ represent the weight of i-th tree in the ensemble and its partial function for the k-th classification respectively.

Finally, the classification of an ensemble of decision tree that adopts voting as its aggregation scheme can be defined as:

$$f(\mathbf{x}) = \mathrm{argmax}_k(f^{(k)}(\mathbf{x}))$$

In this section, we discussed the Fourier representation of decision trees. We showed that the Fourier spectrum of a decision tree is very compact in size. In particular, we proved the exponential decay property is also true for a Fourier spectrum of non-Boolean decision trees. In the next section, we will describe how the Fourier spectrum of an ensemble can be used to construct a single tree.

5 Construction of a Decision Tree from Fourier Spectrum

This section discusses an algorithm to construct a tree from the Fourier spectrum of an ensemble of decision trees. The following section first shows that the information gain needed to choose an attribute at the decision nodes can be efficiently computed from the Fourier coefficients.

5.1 Schema Average and Information Gain

Let us define the *schema average* function as follows:

$$\phi(\mathbf{h}) = \frac{1}{|\mathbf{h}|} \sum_{\mathbf{x} \in \mathbf{h}} f(\mathbf{x}) \tag{6}$$

where $f(\mathbf{x})$ is the classification value of $\mathbf{x}$ and $|\mathbf{h}|$ denotes the number of members in schema $\mathbf{h}$.

Recall that a schema $\mathbf{h}$ denotes a path to a node n_k in a decision tree. Consequently, the average classification value of $\mathbf{h}$ essentially illustrates the classification confidence at n_k (in binary classification problems). $\phi(\mathbf{h})$ can also used to compute the entropy at n_k:

$$\begin{aligned} \text{confidence}(\mathbf{h}) &= \max(\phi(\mathbf{h}), 1 - \phi(\mathbf{h})) \\ \text{entropy}(\mathbf{h}) &= -\phi(\mathbf{h}) \log \phi(\mathbf{h}) - (1 - \phi(\mathbf{h})) \log(1 - \phi(\mathbf{h})) \end{aligned}$$

The computation of $\phi(\mathbf{h})$ using Equation 6 for a given ensemble is not practical since we need to evaluate all $\mathbf{x} \in \mathbf{h}$. Instead we can use the following expression that computes $\phi(\mathbf{h})$ directly from the given FS:

$$\phi(\mathbf{h}) \;=\; \sum_{p_1} .. \sum_{p_m} \exp^{2\pi i (\frac{p_1 b_1}{\lambda_{j_1}} + .. + \frac{p_m b_m}{\lambda_{j_m}})} w_{(0,..,p_1,..,p_m,..,0)} \tag{7}$$

where $\mathbf{h} = ***b_1 *** b_2 *** b_m ***$ that has m non-wildcard values b_i at position j_i and $p_i \in \{0, 1, ..., \lambda_{j_i} - 1\}$. A similar Walsh analysis-based approach for analyzing the behavior of genetic algorithms can be found elsewhere [11].

Using Equation 7 as a tool to obtain information gain, it is relatively easy to come up with a version of ID3 or C4.5-like algorithms that work using the Fourier 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 FS of the ensemble is very compact in size, most FCs involved in computing $\phi(\mathbf{h})$ are zero. Therefore, the evaluation of $\phi(\mathbf{h})$ using Equation 7 is not only inefficient but also involves unnecessary computations.

Construction of a more efficient algorithm to compute $\phi(\mathbf{h})$ is possible by taking advantage of the recursive and decomposable nature of Equation 7. When computing the average of order l schema $\mathbf{h}$, we can reduce some computational steps if any of the order l-1 schemata which subsumes $\mathbf{h}$ is already evaluated. For a simple example in the Boolean domain, let us consider the evaluation of $\phi(*1*0**)$. Let us also assume that $\phi(*1***)$ is pre-calculated. Then, $\phi(*1*0**)$ is obtained by simply adding w_{000100} and $-w_{010100}$ to $\phi(*1***)$. This observation leads us to an efficient algorithm to evaluate schema averages. Recall that the path to a node from the root in a decision tree can be represented as a schema. Then, choosing an attribute for the next node is essentially the same as selecting the best schema among those *candidate* schemata that are subsumed by the current schema and whose orders are just one higher. In the following section, we describe a tree construction algorithm that is based on these observations.

```
Function TCFS(input:  Fourier Spectrum FS)
  Initialize Candidate Feature Set CFSET
  create $root$ node
  $\mathbf{h} \leftarrow$ (***...***)
  $root \leftarrow$ Build($\mathbf{h}$, FS,SFSET)
  return $root$
end
```

Figure 4: Algorithm for constructing a decision tree from Fourier spectrum (TCFS).

5.2 Bottom-up Approach to Construct a Tree

Before describing the algorithm, we need to introduce some notations. Let $\mathbf{h}_{k=i}$ and $\mathbf{h}$ be two schemata. The order of $\mathbf{h}_{k=i}$ is one higher than that of $\mathbf{h}$. Schema $\mathbf{h}_{k=i}$ is identical to $\mathbf{h}$ except at one position—the k-th feature is set to i. For example, consider schemata $\mathbf{h} =$ (*1**2) and $\mathbf{h}_{3=1} = (*1*12)$.

Here we assign an integer number-based ordering among the features (zero for the leftmost feature). $\pi(\mathbf{h})$ denotes a set of partitions that are required to compute $\phi(\mathbf{h})$ (See Equation 7). A *k-fixed partition* is a partition with a non-zero value at the k-th position. Let $\xi(k)$ be a set of order one k-fixed partitions; $\gamma(\mathbf{h}_{k=i})$ be the partial sum of $\phi(\mathbf{h}_{k=i})$ which only includes k-fixed partitions. Now the information gain achieved by choosing the k-th feature with a given $\mathbf{h}$ is redefined using these new notations:

$$\begin{aligned}
\text{Gain}(\mathbf{h}, k) &= \text{entropy}(\mathbf{h}) - \frac{1}{\lambda_k}\sum_{i=0}^{\lambda_k - 1} \text{entropy}(\mathbf{h}_{k=i}) \\
\text{entropy}(\mathbf{h}_{k=i}) &= -\phi(\mathbf{h}_{k=i})\log(\phi(\mathbf{h}_{k=i})) - (1-\phi(\mathbf{h}_{k=i}))\log(1-\phi(\mathbf{h}_{k=i})) \\
\phi(\mathbf{h}_{k=i}) &= \phi(\mathbf{h}) + \gamma(\mathbf{h}_{k=i}) \\
\gamma(\mathbf{h}_{k=i}) &= \sum_{\mathbf{j}\in\pi(\mathbf{h})\otimes\xi(k)} \overline{\psi}_{\mathbf{j}}(\mathbf{h}_{k=i}) w_{\mathbf{j}}
\end{aligned}$$

where $\otimes$ is the Cartesian product and λ_k is the cardinality of the k-th feature, respectively.

Now we are ready to describe the Tree Construction from Fourier Spectrum (TCFS) algorithm, which essentially notes the decomposable definition of $\phi(\mathbf{h}_{k=i})$ and focuses on computing $\gamma(\mathbf{h}_{k=i})$-s. Note that with a given $\mathbf{h}$ (the current path), selecting the next feature is essentially identical to choose the k-th feature that achieves the maximum $Gain(\mathbf{h}, k)$. Therefore, the basic idea of TCFS is to associate most up-to-date $\phi(\mathbf{h}_{k=i})$-s with the k-th feature. In other words, when TCFS selects the next node (after some i is chosen for $\mathbf{h}_k = i$), $\mathbf{h}_{k=i}$ becomes the new $\mathbf{h}$. Then, it identifies a set of FCs (We call these *appropriate* FCs) that are required to compute all $\mathbf{h}_{k=i}$-s for each feature and computes the corresponding entropy. This process can be considered to update each $\phi(\mathbf{h}_{k=i})$ for the corresponding k-th feature as if it were selected. The reason is that such computations are needed anyway if a feature is to be selected in the future along the current path. This is essentially updating $\phi(\mathbf{h}_{k=i})$-s for a feature k using bottom-up approach (following the flavor of dynamic programming). Note that $\phi(\mathbf{h}_{k=i})$ is, in fact, computable by adding $\gamma(\mathbf{h}_{k=i})$ to $\phi(\mathbf{h})$. Here $\gamma(\mathbf{h}_{k=i})$-s are partial sums that only current appropriate FCs contribute. Detection of all appropriate FCs requires a scan over the FS. However, they are split from the FS once they are used in computation, since they are no longer needed for the calculation of higher order schemata. Thus it takes a lot less time to compute higher order schemata; note that it is just opposite to what we encountered in the naive implementation. The algorithm stops growing a path when either the original FS becomes an empty set or the minimum confidence level is achieved. The depth of the resulting tree can be set to a pre-determined bound. A pictorial description of the algorithm is shown in Figure 6. Pseudo code of the algorithm is presented in Figures 4 and 5.

The TCFS uses the same criteria to construct a tree as that of the C4.5. Both of them require a number of information-gain-tests that grows exponentially with respect to the depth of the tree. In that sense, the asymptotic running time of TCFS is the same as that of the C4.5. However, while the C4.5 uses original data to compute information gains, TCFS uses a Fourier spectrum. Therefore, in practice, a comparison of the running time between the two approaches will depend on the sizes of the original data and that of Fourier spectrum. The following section presents an extension of the TCFS for handling non-Boolean class labels.

5.3 Extension of TCFS to Multi-Class Decision Trees

The extension of TCFS algorithm to multi-class problems is immediately possible by redefining the "entropy" function. It should be modified to capture an entropy from the multiple class labels. For this, let us first define $\phi^{(i)}(\mathbf{h})$ to be a schema average function that uses $\Im_i$ (See Section 4.4) only. Note that it computes the average occurence of the i-th class label in $\mathbf{h}$. Then the entropy of a schema is redefined as follows.

$$\text{entropy}(\mathbf{h}) = -\sum_{i=1}^{k} \phi^{(i)}(\mathbf{h}) \log \phi^{(i)}(\mathbf{h})$$

where k is the number of class labels.

This expression can be directly used for computing the information gain to choose the decision nodes in a tree for classifying domains with non-Boolean class labels.

In this section, we discussed a way to assign a confidence to a node in a decision tree, and considered a method to estimate information gain using it. Consequently, we showed that a decision tree construction from the Fourier spectrum is possible. In particular, we devised TCFS algorithm that exploits the recursive and decomposable nature of tree building process in spectrum domain, thus constructing a decision tree efficiently. In the following section, we will discuss empirical verification of the proposed Fourier spectrum-based aggregation approach.

6 Empirical Study of Ensemble Models with Fourier Analysis

This section reports the experimental performance of the proposed decision tree aggregation scheme using a semi-synthetic data stream with 100 discrete attributes. The objective is to continuously evolve a decision tree-based predictive model for a particular Boolean attribute. The data-stream generator is essentially a C4.5 decision tree learned from three years of Nasdaq 100 stock quote data. The original data is pre-processed and transformed to discrete data by encoding percentages of changes in stock quotes between consecutive days. For these experiments, we assigned 4 discrete values that denote levels of changes. Decision trees predict whether the Yahoo stock is likely to increase or decrease based on the attribute values of the 99 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 distribution. This distribution is changed many times during a single experiment. We also added white noise to the generator. A test data set of 10,000 instances is generated from the data source in advance to the experiment.

6.1 Classification Accuracy of Naive Bagging and Arcing Based Model

We implemented versions of BEM (naive Bagging), Arcing and AdaBoost suitable for a stream data mining environment as discussed in the previous section. We performed various tests over the validation data which are described above. We studied the accuracy of each model with various sizes (N) of data blocks at each update. We used $N = 100$, 200, 300, 400 and 500. All the results were measured out of 100 iterations (or updates). We performed tests on all three implementations of ensemble models, BEM (Bagging), Arcing and AdaBoost. However, we only report the results of Bagging and Arcing here since the AdaBoost did not perform well, compared to the other two.

Figure 7 plots the classification accuracies of Bagging and Arcing with various data block sizes. It also compares the classification accuracies that are measured with the Fourier spectrum of each model in Figure 8. The graph indicates that the two representations (tree ensemble and its Fourier spectrum) are functionally equivalent.

6.2 Fourier Spectra of Ensemble Models: Experimental Results

This section shows the distribution of energy over each $\mathcal{F}_k$ (all non-zero Fourier coefficients of order k),

$$E(\mathcal{F}_k) \;=\; \sum_{w_{\mathbf{i}} \in \mathcal{F}_k} ||w_{\mathbf{i}}||^2$$

where $||\cdot||$ denotes the magnitude (L_2 norm) of $w_{\mathbf{i}}$.

Figure 9 (Left) shows distribution of the first five $E(\mathcal{F}_k)$ for both Bagging and Arcing ensemble models discussed in the previous section. All decision trees in the original ensemble models have a depth of five or six, which bounds the maximum Fourier Coefficient (FC) order to six. However, as clearly demonstrated in the figure, most of the information (or energy) is concentrated at lower order $\mathcal{F}_k$-s. The average magnitude of FCs at each order is plotted in Figure 9 (Right). The graph also shows rapid decrease of energy as order grows. It faithfully demonstrates the exponential decay property as in Fourier spectrum of a single decision tree.

Block Size	FCs (Bagging)	Decision Nodes (Bagging)	FCs (Arcing)	Decision Nodes (Arcing)
100	103	3008	151	3300
200	148	4288	322	5216
300	256	4904	688	6092
400	295	5544	1075	6836
500	319	6096	1336	7468

Table 1: 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 that are learned with various block sizes.

To see how the energy, preserved in each $\mathcal{F}_k$, affects the classification, we first performed a 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. This is essentially to discover a small subset of FS which is sufficient for classification. Classification was done using the inverse Fourier transform. The experiment was conducted with the same data set mentioned previously. Also, it was restricted to a data block size of 500. Figures 10 compare classification accuracy of each subset of the Fourier spectrum in comparison with regular decision tree ensemble models which are learned with Bagging and Arcing respectively. Even with $\mathcal{F}_0, \mathcal{F}_1$ and $\mathcal{F}_2$, we observed decent classification accuracy. The difference from regular ensemble models became negligible when $\mathcal{F}_3$ was included in the classification. This is indeed an empirical verification of exponential decay property of Fourier Coefficients in terms of classification accuracy.

We also studied the size of Fourier spectra of ensemble models in terms of the number of FCs. We noted that the size of Fourier spectra can be reduced significantly by neglecting any FC whose magnitude is small. For this, we sorted the FCs, which are included in $\mathcal{F}_0, \ldots, \mathcal{F}_5$, in the descending order by their magnitudes and plotted accumulated energy distribution. Figures 11 show these graphs. For both Bagging and Arcing models, most of the energy is preserved in a small percentage of FCs (from the top), which indicates that most FCs have close-to-zero magnitudes. It should be noted that percentages are defined over FCs we extracted from 100 trees, not over the entire FCs (in the complete Fourier domain). We also charted classification accuracy versus percentage of FCs (from the top) and observed that less than two percent of the FCs are sufficient for classification (See Figures 12). 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, two percent of the all FCs is a significant reduction in representing an ensemble model. Table 1 compares the number of nodes in an ensemble of decision trees and the numbers of the FCs when approximately the same classification accuracy was observed. The table shows that for each block size in the current experimental set up, the number of FCs in the Fourier spectrum of the ensemble genrated by Arcing is larger than that constructed by Bagging. The following section presents experimental results regarding the performance of the tree construction algorithm.

Type	Block Size	Average	Ensemble	Fourier Tree
Bagging	100	79.45(%)	88.68(%)	88.11(%)
	200	89.68(%)	94.11(%)	92.75(%)
	300	92.67(%)	95.57(%)	94.82(%)
Arc-fx	100	73.18(%)	92.26(%)	91.83(%)
	200	87.75(%)	96.21(%)	95.28(%)
	300	90.98(%)	97.78(%)	96.32(%)

Table 2: Accuracies of ensembles and Fourier trees. *Average* stands for the average accuracy of individual base model in an ensemble.

Type	Block	Confidence		
	Size	0.5-0.6	0.6-0.8	0.8-1.0
Bagging	100	71.11(%)	71.04(%)	100(%)
	200	63.52(%)	67.65(%)	100(%)
	300	64.01(%)	75.1(%)	99.1(%)
Arc-fx	100	73.08(%)	100(%)	100(%)
	200	78.72(%)	99.5(%)	100(%)
	300	75.33(%)	100(%)	100(%)

Table 3: The accuracies leaf nodes of different confidence levels.

6.3 Tree Construction using TCFS

This section reports the results of the proposed TCFS algorithm. TCFS is applied to construct decision trees from Fourier spectra of Bagging and Arc-fx ensembles described in Table 1. In particular, we report the results of N=100, 200 and 300.

Table 2 compares the classification accuracies of the original ensemble models and the trees that are constructed from their aggregated Fourier spectrum. We call these *Fourier Trees*. Also, the average accuracy of an individual tree in an ensemble is shown in Table 2. Since most trees in each ensemble have depths of less than 5, we fixed the depth of each Fourier tree to 5. We also set the minimum confidence level (for early stopping criteria) to 0.9. Interestingly enough, we could construct Fourier trees that are comparable to the original ensembles.

Table 3 shows how the confidence associated with each leaf node in the aggregated tree affects the accuracy. For this we divide confidence ranges into three groups; [0.5,0.6), [0.6,0.8) and [0.8,1.0]. We then measured the average accuracy performance of each group. The results clearly convey the idea that leaf nodes with high confidences tend to be more accurate. This confirms that our original intention of extracting descriptive patterns is valid. One interesting observation is that a leaf with a relatively low confidence produces a high accuracy in Arc-fx, which is not the case in Bagging.

In this section, we presented the empirical analysis of the Fourier spectra of two ensemble models and the trees constructed from them. In particular, we studied how the energy distribution of each model in the Fourier domain affects the overall performance (e.g., accuracy and storage overhead). Various experimental results reported here strongly confirm that a complex ensemble model can be reduced to a very compact Fourier spectrum. Various experimental results reported here strongly confirm that a complex ensemble model can be reduced to a very compact Fourier spectrum and a tree constructed from it. Particularly, we illustrated comparable accuracy performances of the Fourier trees. We also showed that confidences associated with each node can be effectively used as a measure to extract significant patterns. Although we did not measure the precise time complexity, detecting and extracting significant FCs from a decision tree is very fast; we observed it takes less than a second to extract significant FCs from a decision tree of around 150 nodes on a Pentium III 1 GHz PC.

7 Conclusions

This paper offers several new developments in our research on the Fourier spectrum of decision trees for mining data streams in a mobile and distributed environment. This particular work however considers the problem of mining data streams using the Fourier spectrum of decision tree ensembles without referring to its mobile and distributed applications.

This paper presents a novel approach to construct a single decision tree from an ensemble of trees. An ensemble model often includes a large number of base models (decision trees in our case). This is true for applications involving either data streams or offline databases. Particularly in data stream environments, data stacks up quickly. So we need to maintain an up-to-date model in a very compact form. Furthermore, a complex ensemble model does not lend itself to knowledge discovery in a human-friendly format due to its complex internal structure. Reducing a complex ensemble model into a simple tractable entity is a prerequisite step toward extracting the underlying knowledge. In this paper, we proposed a Fourier spectrum approach that addresses these issues.

We showed that the information gain can be directly computed from the Fourier spectrum of an ensemble classifier. Subsequently, a fast algorithm that constructs a tree from the spectrum was proposed. Since the structure of an ensemble classifier is believed to be complex, we specifically proposed a measure to extract significant patterns out of the aggregated tree. We extended the technique for dealing with domains with non-Boolean class labels. We studied both analytical and practical aspects of the proposed technique. We also proved the exponential energy-decay-property of the Fourier spectrum of a decision tree with non-Boolean categorical attributes.

The Fourier representation-based approach presented in this paper is applicable to categorical data. If the features are continuous then we first need to discretize them before learning the tree. We need to explore techniques for constructing the spectrum of the trees that dynamically discretize continuous features while building the tree, resulting in different discretizations in different portions of the tree. We plan to explore this in the future.

Acknowledgments

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

References

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

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

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

[4] Walter Van de Velde. Incremental induction of topologically minimal trees. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 66–74, San Mateo, CA, 1990. Morgan Kaufmann.

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

[6] Pedro Domings and Geoff Hulten. Mining high-speed data streams. In *Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Boston, MA, August, 2000.

[7] Harris Drucker and Corrina Cortes. Boosting decision trees. *Advances in Neural Information Processing Systems*, 8:479–485, 1996.

[8] Wei Fan, Sal Stolfo, and Junxin 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.

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

[10] Johannes Gehrke, Venkatesh Ganti, Raghu Ramakrishnan, and Wei-Yin Loh. BOAT — optimistic decision tree construction. In *Proceedings of SIGMOD, ACM*, pages 169–180, 1999.

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

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

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

[14] H. Kargupta and B. Park. Mining time-critical data stream using the Fourier spectrum of decision trees. In *Proceedings of the IEEE International Conference on Data Mining*, pages 281–288. IEEE Press, 2001.

[15] H. Kargupta, B.H. Park, S. Pittie, L. Liu, D. Kushraj, and K. Sarkar. Mobimine: Monitoring the stock market from a PDA. *ACM SIGKDD Explorations*, 3(2):37–46, January 2002.

[16] E. Kushilevitz and Y. Mansour. Learning decision trees using the Fourier spectrum. *SIAM Journal oo Computing*, 22(6):1331–1348, 1993.

[17] S. W. Kwok and C. Carter. Multiple decision trees. *Uncertainty in Artificial Intelligence 4*, pages 327–335, 1990.

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

[19] 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.

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

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

[22] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.

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

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

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

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

[27] Jeffrey C. Schlimmer and Richard Granger Jr. Beyond incremental processing: Tracking concept drift. In *AAAI, Vol. 1*, pages 502–507, 1986.

[28] W. Nick Street and YongSeog Kim. A streaming ensemble algorithm (sea) for large-scale classificaiton. In *Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, 2001.

[29] P. E. 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.

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

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

Appendix: The Exponential Decay Property of the non-Boolean Fourier Spectrum

Let us consider l-dimensional discrete domain $\Lambda = \prod_{j=1}^{l}\{0, ..., \lambda_j - 1\}$, where each element $\mathbf{x} \in \Lambda$ is denoted as $(x_1, x_2, ..., x_l)$. Note that $\lambda_1, \lambda_2, \cdots, \lambda_l$ denote the cardinalities of $x_1, x_2, ..., x_l$ respectively. That is, the component x_j can take only the values $0, 1, \ldots, \lambda_j - 1$.

The Fourier basis function that corresponds to the partition $\mathbf{j}$ is,

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

The Fourier transform that corresponds to the partition $\mathbf{j}$ is then,

$$w_{\mathbf{j}} = \Pi_{i=1}^{l} \frac{1}{\lambda_i} \sum_{\mathbf{x}} \psi_{\mathbf{j}}^{\overline{\lambda}}(\mathbf{x}) \phi(\mathbf{x})$$

The inverse Fourier transform for an instance vector $\mathbf{x}$ is,

$$f(\mathbf{x}) = \sum_{\mathbf{j}} w_{\mathbf{j}} \overline{\psi}_{\mathbf{j}}^{\overline{\lambda}}(\mathbf{x})$$

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

Now we prove the exponential decay property of Fourier spectrum in a non-Boolean domain. For the sake of simplicity, let us further assume that each $\lambda_j = 2^{q_j}$, where q_j is any non-negative integer, so that the j-th non-binary variable can be represented using q_j bits. The results stated here are general and they extend easily to the case where the cardinalities are not powers of two. In the following proof, we treat a schema[1] as both a subset of Λ and a string in $\overline{\Lambda} = \prod_{j=1}^{l}\{*, 0, ..., \lambda_j - 1\}$. $\overline{\Lambda}$ is the set of all schemata on Λ. The proof is based on transforming each non-Boolean string in the l-dimensional space Λ to a (longer) Boolean string in $\Lambda' = \{0, 1\}^Q$, where $Q = \sum_{j=1}^{l} q_j$. This is done by replacing each feature of a string in Λ with its binary expansion. The following arguments establish a correspondence between the energy contained in corresponding sets of Fourier coefficients in the spectrum of the function defined on Λ and the function induced on Λ' by the transformation. Thus, since the exponential decay property holds for the Boolean case, it must hold for the general discrete case as well.

We next define the transformation

$$\kappa : \overline{\Lambda} \mapsto \overline{\Lambda'},$$

formally. $\overline{\Lambda'}$ refers to the set of all schemata on Λ', that is, $\overline{\Lambda'} = \{*, 0, 1\}^Q$. We first define feature-wise transformations

$$b_j : \{*, 0, 1, ..., \lambda_j - 1\} \mapsto \{*, 0, 1\}^{q_j}$$

[1]A schema is a hyperplane that denotes a subset of domain members. It is essentially similarity based equivalence class. Schemata is its plural form.

by

$$b_j(x_j) = \begin{cases} \overbrace{**...*}^{q_i} & \text{if } x_j = * \\ x'_j & \text{if } x_j \in \{0, 1, ..., \lambda_j - 1\} \end{cases}$$

where x'_j is the q_j-bit binary representation of x_j. Now for any schema $\mathbf{s} = (s_1, s_2, ..., s_l) \in \overline{\Lambda}$, κ is defined as

$$\kappa : (s_1, s_2, ..., s_l) \mapsto (b_1(s_1), b_2(s_2), ..., b_l(s_l))$$

κ is essentially a map from an l-feature schema in an arbitrary discrete domain to a Q-feature schema in a binary domain. We note here that we treat Λ as a subset of $\overline{\Lambda}$ and thus can apply κ to elements of Λ. For a subset $A \subseteq \Lambda$ we use the notation $\kappa(A)$ to denote the set $\{\kappa(\mathbf{x}) \mid \mathbf{x} \in A\}$.

Let us further assume that $f(\mathbf{x})$ is a functional representation of a decision tree T whose domain of definition is Λ. We establish some further before we proceed. We use Γ to denote the set of *schemata* $\{*, 0\}^l$. That is, schemata in Γ have only zeroes at their fixed (non-wildcard) features. We also define a set of schemata associated with any fixed schema $\mathbf{s} \in \Gamma$:

$$S(\mathbf{s}) = \{\mathbf{h} \in \overline{\Lambda} \mid h_k = *, \text{ if } s_k = \ 0 \text{ and } h_k \in \{0, 1, ..., \lambda_k - 1\}, \text{ otherwise}\}$$

where h_k denotes the k-th feature in the schema $\mathbf{h}$, and similarly for s_k. Thus elements of $S(\mathbf{s})$ can have wildcards at those positions where $\mathbf{s}$ has zeroes.

Lemma 5 *For any schema* $\mathbf{s} \in \Gamma$,

$$\sum_{\mathbf{i} \in \mathbf{s}} \eta_{\mathbf{i}} \overline{\psi}_{\mathbf{i}}^{\overline{\lambda}}(\mathbf{x}) = \sum_{\mathbf{j} \in \kappa(\mathbf{s})} w_{\mathbf{j}} \psi_{\mathbf{j}}(\kappa(\mathbf{x}))$$

Proof *For any* $\mathbf{i} \in \mathbf{s}$,

$$\begin{aligned} \eta_{\mathbf{i}} &= \frac{1}{|\Lambda|} \sum_{\mathbf{x} \in \Lambda} f(\mathbf{x}) \psi_{\mathbf{i}}^{\overline{\lambda}}(\mathbf{x}) \\ &= \frac{1}{|S(\mathbf{s})|} \sum_{\mathbf{h} \in S(\mathbf{s})} \frac{1}{|\mathbf{h}|} \sum_{\mathbf{x} \in \mathbf{h}} f(\mathbf{x}) \psi_{\mathbf{i}}^{\overline{\lambda}}(\mathbf{x}) \end{aligned}$$

where $|\mathbf{h}|$ *and* $|S(\mathbf{s})|$ *denote the sizes of* $\mathbf{h}$ *and* $S(\mathbf{s})$ *respectively. Now for any* $\mathbf{i} \in \mathbf{s}$ *and* $\mathbf{h} \in S(\mathbf{s})$, $\overline{\psi}_{\mathbf{i}}^{\overline{\lambda}}(\mathbf{x})$ *is invariant over* $\mathbf{x} \in \mathbf{h}$. *Let us denote this value by* $\overline{\psi}_{\mathbf{i}}^{\overline{\lambda}}(\mathbf{h})$. *Then we get*

$$\begin{aligned} \eta_{\mathbf{i}} &= \frac{1}{|S(\mathbf{s})|} \sum_{\mathbf{h} \in S(\mathbf{s})} \frac{1}{|\mathbf{h}|} \sum_{\mathbf{x} \in \mathbf{h}} f(\mathbf{x}) \psi_{\mathbf{i}}^{\overline{\lambda}}(\mathbf{x}) \\ &= \frac{1}{|S(\mathbf{s})|} \sum_{\mathbf{h} \in S(\mathbf{s})} \frac{\psi_{\mathbf{i}}^{\overline{\lambda}}(\mathbf{h})}{|\mathbf{h}|} \sum_{\mathbf{x} \in \mathbf{h}} f(\mathbf{x}) \\ &= \frac{1}{|S(\mathbf{s})|} \sum_{\mathbf{h} \in S(\mathbf{s})} \phi(\mathbf{h}) \psi_{\mathbf{i}}^{\overline{\lambda}}(\mathbf{h}) \end{aligned}$$

where $\phi(\mathbf{h})$ is the average of $f(\mathbf{x})$, for $\mathbf{x} \in \mathbf{h}$. Now $\phi(\mathbf{h})$ (by inverse Fourier transform) for any $\mathbf{h} \in S(\mathbf{s})$ is

$$\phi(\mathbf{h}) = \frac{1}{|\mathbf{h}|} \sum_{x \in \mathbf{h}} \sum_{\mathbf{i} \in \Lambda} \eta_{\mathbf{i}} \overline{\psi}_{\mathbf{i}}^{\lambda}(\mathbf{x}) = \frac{1}{|\mathbf{h}|} \sum_{\mathbf{i} \in \Lambda} \eta_{\mathbf{i}} \sum_{x \in \mathbf{h}} \overline{\psi}_{\mathbf{i}}^{\lambda}(\mathbf{x}) = \sum_{\mathbf{i} \in \mathbf{s}} \eta_{\mathbf{i}} \overline{\psi}_{\mathbf{i}}^{\lambda}(\mathbf{h})$$

since $\sum_{x \in \mathbf{h}} \overline{\psi}_{\mathbf{i}}^{\lambda}(\mathbf{h})$ equals 0 if $\mathbf{i} \notin \mathbf{s}$ and $|\mathbf{h}|$ otherwise. Similarly, $\phi(\kappa(\mathbf{h}))$ for any $\mathbf{h} \in S(\mathbf{s})$ is

$$\phi(\kappa(\mathbf{h})) = \sum_{\mathbf{j} \in \mathbf{s}} w_{\mathbf{j}} \psi_{\mathbf{j}}(\kappa(\mathbf{h}))$$

Since for any $\mathbf{h}$, $\phi(\mathbf{h}) = \phi(\kappa(\mathbf{h}))$,

$$\sum_{\mathbf{i} \in \mathbf{s}} \eta_{\mathbf{i}} \overline{\psi}_{\mathbf{i}}^{\lambda}(\mathbf{h}) = \sum_{\mathbf{j} \in \kappa(s)} w_{\mathbf{j}} \psi_{\mathbf{j}}(\kappa(\mathbf{h}))$$

Therefore,

$$\sum_{\mathbf{i} \in \mathbf{s}} \eta_{\mathbf{i}} \overline{\psi}_{\mathbf{i}}^{\lambda}(\mathbf{x}) = \sum_{\mathbf{j} \in \kappa(\mathbf{s})} w_{\mathbf{j}} \psi_{\mathbf{j}}(\kappa(\mathbf{x})), \text{ for all } \mathbf{x} \in \mathbf{h}.$$

This completes the proof of the lemma. ■

Now let us define $\theta(\mathbf{s})$ as,

$$\theta(\mathbf{s}) = \{\mathbf{i} \in \mathbf{s} \mid o(\mathbf{i}) = o(\mathbf{s})\}$$

where $o(\mathbf{i})$ and $o(\mathbf{s})$ denote the orders of $\mathbf{i}$ and $\mathbf{s}$ respectively. $\theta(\mathbf{s})$ is a subset of $\mathbf{s}$ which only includes partitions whose orders are the same as that of $\mathbf{s}$. Now consider the following corollary.

Corollary 1 *For any $\theta(\mathbf{s})$ and $\kappa(\theta(\mathbf{s}))$,*

$$\sum_{\mathbf{i} \in \theta(\mathbf{s})} \eta_{\mathbf{i}} \overline{\psi}_{\mathbf{i}}^{\lambda}(\mathbf{x}) = \sum_{\mathbf{j} \in \kappa(\theta(\mathbf{s}))} w_{\mathbf{j}} \psi_{\mathbf{j}}(\kappa(\mathbf{x}))$$

Proof *Let $P_{\mathbf{s},n}$ be the set of all schemata that are obtained by replacing n of the *-s in $\mathbf{s}$ with zero (there are $\binom{l-o(\mathbf{s})}{n}$ such strings). Then,*

$$\begin{aligned}
\sum_{\mathbf{i} \in \theta(\mathbf{s})} \eta_{\mathbf{i}} \psi_{\mathbf{i}}(\mathbf{x}) \overline{\psi}_{\mathbf{i}}^{\lambda}(\mathbf{x}) &= \sum_{\mathbf{k} \in \mathbf{s}} \eta_{\mathbf{k}} \overline{\psi}_{\mathbf{k}}^{\lambda}(\mathbf{x}) + \sum_{n=1}^{o(\mathbf{s})} (-1)^n \sum_{\mathbf{r} \in P_{\mathbf{s},n}} \sum_{\mathbf{k} \in \mathbf{r}} \eta_{\mathbf{k}} \overline{\psi}_{\mathbf{k}}^{\lambda}(\mathbf{x}) \\
&= \sum_{\mathbf{m} \in \kappa(\mathbf{s})} w_{\mathbf{m}} \psi_{\mathbf{m}}(\kappa(\mathbf{x})) + \sum_{n=1}^{o(\mathbf{s})} (-1)^n \sum_{\mathbf{r} \in P_{\mathbf{s},n}} \sum_{\mathbf{m} \in \kappa(\mathbf{r})} w_{\mathbf{m}} \psi_{\mathbf{m}}(\kappa(\mathbf{x})) \\
&= \sum_{\mathbf{j} \in \kappa(\theta(\mathbf{s}))} w_{\mathbf{j}} \psi_{\mathbf{j}}(\kappa(\mathbf{x}))
\end{aligned}$$

The second equality follows from Lemma 5. ■

Now let us rephrase Linial's original lemma as follows. That is, in the Boolean Fourier basis notation,

$$\sum_{o(\mathbf{j})\geq k} w_{\mathbf{j}}^2 \leq \wp(k)$$

where $\wp(k)$ decreases exponentially in k. Now consider the main lemma.

Lemma 6 *In a non-Boolean l-dimensional discrete domain $\mathbf{D}$, for any non-negative integer $k \leq l$ and Fourier spectrum $\eta_{\mathbf{i}}$-s of a decision tree defined over $\mathbf{D}$,*

$$\sum_{o(\mathbf{i})\geq k} ||\eta_{\mathbf{i}}||^2 \leq \wp(k)$$

where $||\eta_{\mathbf{i}}||$ denotes the magnitude of $\eta_{\mathbf{i}}$.

Proof: *For the sake of convenience, let us define,*

$$\begin{aligned} f_{\mathbf{s}}(\mathbf{x}) &= \sum_{\mathbf{i}\in\theta(\mathbf{s})} \eta_{\mathbf{i}}\overline{\psi}_{\mathbf{i}}^{\lambda}(\mathbf{x}) \\ f_{\kappa(s)}(\kappa(\mathbf{x})) &= \sum_{\mathbf{j}\in\kappa(\theta(\mathbf{s}))} w_{\mathbf{j}}\psi_{\mathbf{j}}(\mathbf{x}) \end{aligned}$$

Then, by Corollary 1, $f_{\mathbf{s}}(\mathbf{x}) = f_{\kappa(\mathbf{s})}(\kappa(\mathbf{x}))$. Following Parseval's Identity,

$$\begin{aligned} \frac{1}{|\Lambda|}\sum_{\mathbf{x}\in\Lambda} f_{\mathbf{s}}^2(\mathbf{x}) &= \frac{1}{|\Lambda|}\sum_{\mathbf{x}\in\Lambda} f_{\kappa(\mathbf{s})}^2(\kappa(\mathbf{x})) \\ &= \sum_{\mathbf{i}\in\theta(\mathbf{s})} ||\eta_{\mathbf{i}}||^2 \\ &= \sum_{\mathbf{j}\in\kappa(\theta(\mathbf{s}))} w_{\mathbf{j}}^2 \end{aligned}$$

Since, for any $\mathbf{i} \in \mathbf{s}$ and $\mathbf{j} \in \kappa(\mathbf{s})$, $o(\mathbf{i}) \leq o(\mathbf{j})$,

$$\begin{aligned} \sum_{o(\mathbf{s})\geq k}\sum_{\mathbf{i}\in\theta(\mathbf{s})} ||\eta_{\mathbf{i}}||^2 &= \sum_{\kappa(\mathbf{s})}\sum_{\mathbf{j}\in\kappa(\theta(\mathbf{s}))} w_{\mathbf{j}}^2 \\ &\leq \sum_{o(\mathbf{j})\geq k} w_{\mathbf{j}}^2 \\ &\leq \wp(k) \end{aligned}$$

Thus, the non-Boolean Fourier spectrum of a decision tree also has the exponential decay property. ■

```
Function Build(input:  Schema $\mathbf{h}$,
Fourier Spectrum $\mathbf{FS}$, Candidate Feature Set $\mathbf{CFSET}$)
  create $root$ node
  odr $\leftarrow (1, order(\mathbf{h}) + 1)$
  $\mathbf{Marked} \leftarrow \phi$
  for each Fourier Coefficient $w_{\mathbf{i}}$ within odr from FS
    ft = intersect($\mathbf{h}$,$\mathbf{i}$,$\mathbf{CFSET}$)
    if ft is not $\phi$
      for each value $j$ of ft
        update $\gamma(\mathbf{h}_{ft=j})$ with $w_i$
      end
      add $w_{\mathbf{i}}$ to $\mathbf{Marked}$
    end
  end
  if $\mathbf{Marked}$ is $\phi$
    set label for $root$ using average of $\mathbf{h}$
    return $root$
  end
  for each feature $f_i$ in $\mathbf{CFSET}$
    $gain_i \leftarrow Gain(\mathbf{h}, f_i)$
  end
  remove $k$ with the maximum $gain_i$ from $\mathbf{CFSET}$
  $root \leftarrow k$
  $\mathbf{FS} \leftarrow \mathbf{FS}$ - $\mathbf{Marked}$
  for each possible branch $br_i$ of $k$
    $\mathbf{h}_{\mathbf{k=i}} \leftarrow$ update $\mathbf{h}$ with $k = i$
    $br_i \leftarrow$ Build($\mathbf{h}_{k=i}$,$\mathbf{FS}$, $\mathbf{CFSET}$)
  end
  add $k$ into $\mathbf{CFSET}$
  add $\mathbf{Marked}$ into $\mathbf{FS}$
  return $root$
end
```

Figure 5: Algorithm for constructing a decision tree from Fourier spectrum (TCFS). order($\mathbf{h}$) returns the order of schema $\mathbf{h}$. intersect($\mathbf{h}$, $\mathbf{i}$) returns the feature to be updated using $w_{\mathbf{i}}$, if such a feature exists. Otherwise it returns ϕ.

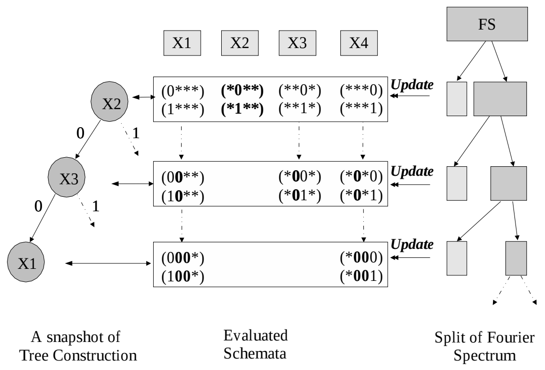

Figure 6: Illustration of the Tree Construction from Fourier Spectrum (TCFS) algorithm

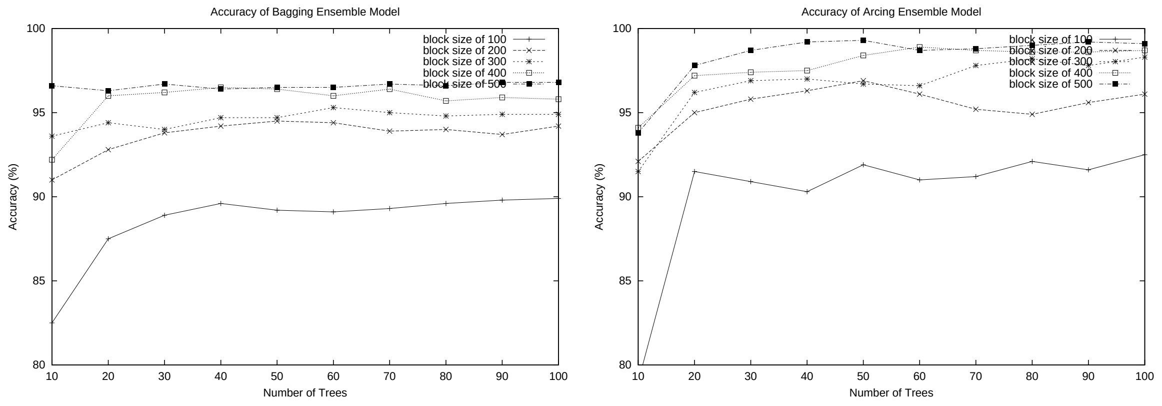

Figure 7: Classification accuracies of Bagging (Left) and Arcing Ensembles (Right) with various block sizes.

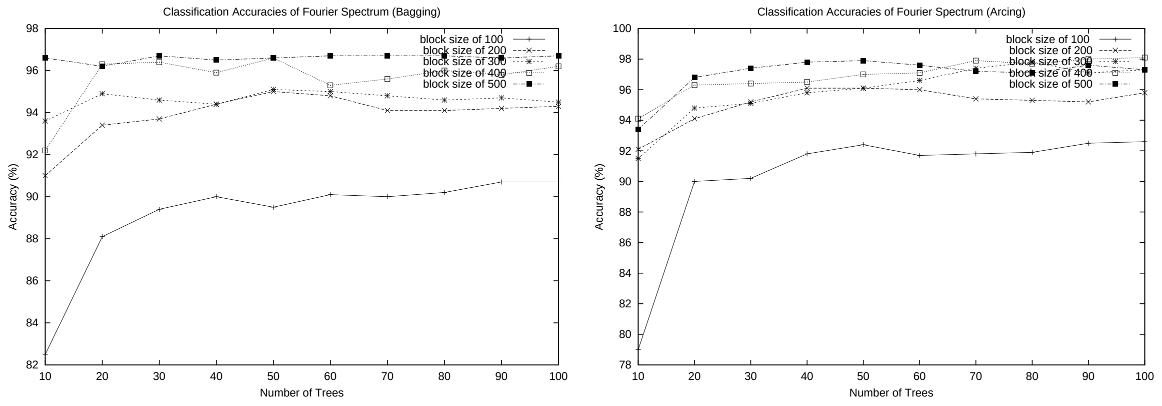

Figure 8: Classification accuracies of Bagging (Left) and Arcing Ensembles (Right) with various block sizes using their corresponding Fourier Spectra.

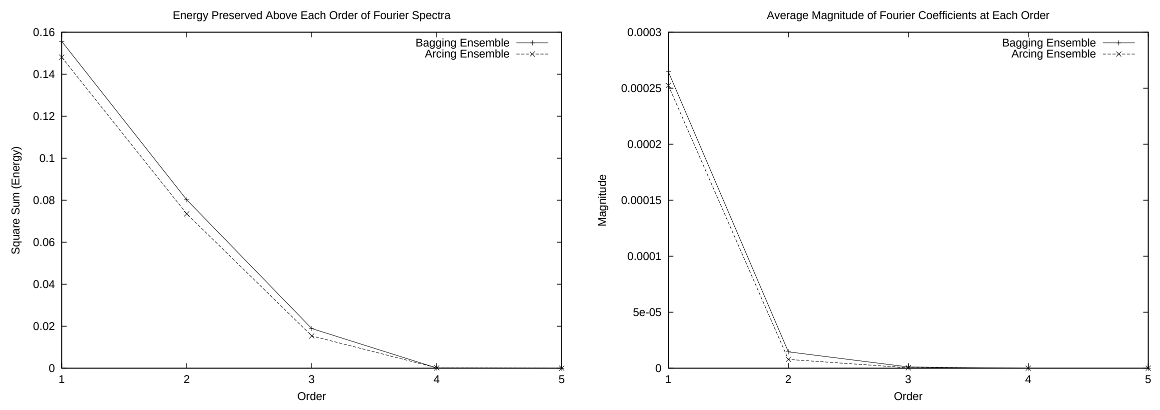

Figure 9: Energy preserved Above each order of Fourier Spectra. (Left) Average magnitude of Fourier coefficients at each order. (Right) Each graph shows energy distribution of ensemble of 100 decision trees for Bagging and Arcing with block size of 500.

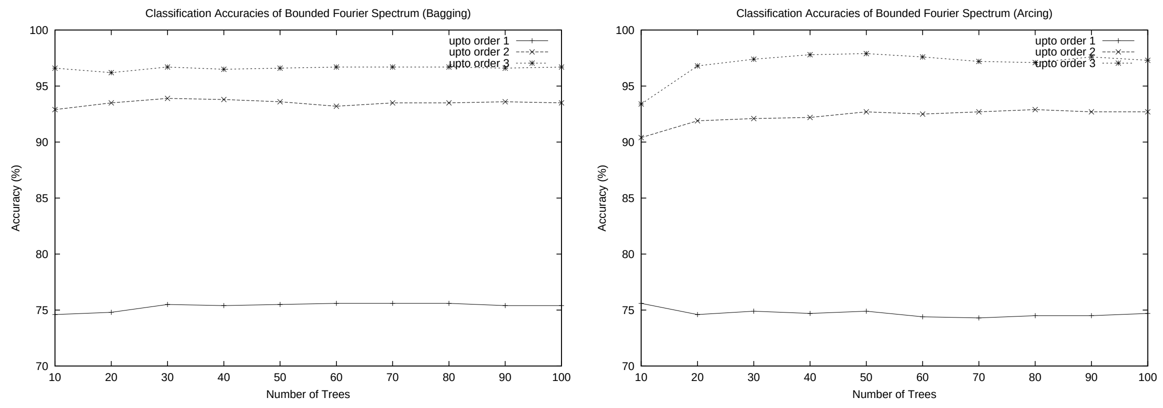

Figure 10: Classification accuracy of Fourier spectra as classifier when FC order is bounded for both Bagging (Left) and Arcing (Right).

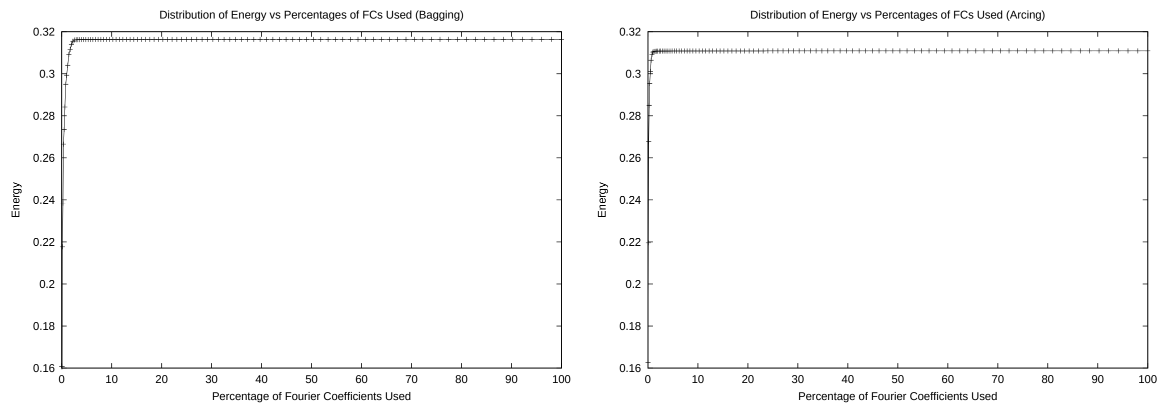

Figure 11: Distribution of energy vs percentage of FCs from the sorted $\{\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \mathcal{F}_4, \mathcal{F}_5\}$ for Bagging (Left) and Arcing (Right).

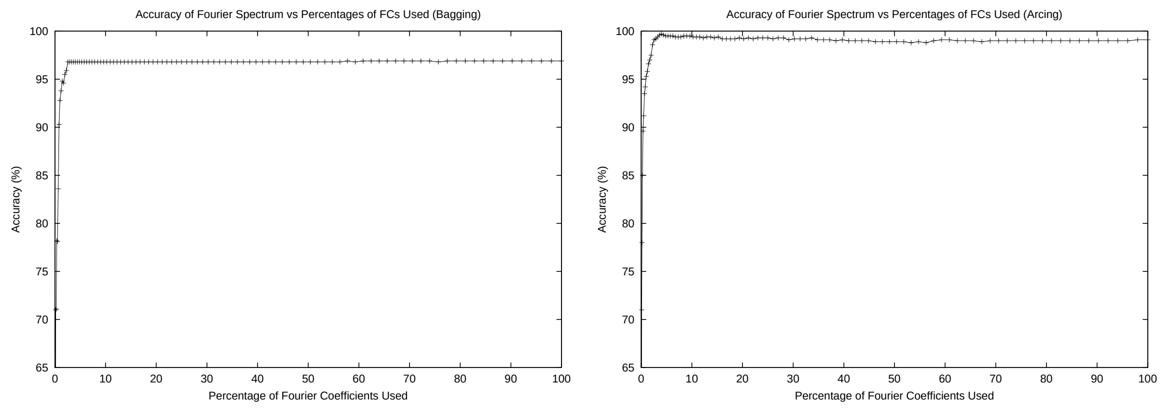

Figure 12: Classification accuracies vs percentage of FCs from the sorted $\{\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \mathcal{F}_4, \mathcal{F}_5\}$ for both Bagging (Left) and Arcing (Right).