Orthogonal Decision Trees

Hillol Kargupta, Byung-Hoon Park, and Haimonti Dutta

Abstract—This paper introduces orthogonal decision trees that offer an effective way to construct a redundancy-free, accurate, and meaningful representation of large decision-tree-ensembles often created by popular techniques such as Bagging, Boosting, Random Forests, and many distributed and data stream mining algorithms. Orthogonal decision trees are functionally orthogonal to each other and they correspond to the principal components of the underlying function space. This paper offers a technique to construct such trees based on the Fourier transformation of decision trees and eigen-analysis of the ensemble in the Fourier representation. It offers experimental results to document the performance of orthogonal trees on the grounds of accuracy and model complexity.

Index Terms—Orthogonal decision trees, redundancy free trees, principle component analysis, Fourier transform.

1 INTRODUCTION

Decision tree [1] ensembles are frequently used in data mining and machine learning applications. Boosting [2], [3], Bagging [4], Stacking [5], and Random Forests [6] are some of the well-known ensemble-learning techniques. Many of these techniques often produce large ensembles that combine the outputs of a large number of trees for producing the overall output. Ensemble-based classification and outlier detection techniques are also frequently used in mining continuous data streams [7], [8]. Large ensembles pose several problems to a data miner. They are difficult to understand and the overall functional structure of the ensemble is not very "actionable" since it is difficult to manually combine the physical meaning of different trees in order to produce a simplified set of rules that can be used in practice. Moreover, in many time-critical applications, such as monitoring data streams in resource-constrained environments [9], maintaining a large ensemble and using it for continuous monitoring are computationally challenging. So, it will be useful if we can develop a technique to construct a redundancy-free meaningful compact representation of large ensembles. This paper offers a technique to do that and possibly more.

This paper presents a technique to construct redundancy-free decision-tree-ensembles by using orthogonal decision trees. The technique first constructs an algebraic representation of trees using multivariate discrete Fourier basis set. The new representation is then used for eigen-analysis of the covariance matrix generated by the decision trees in Fourier representation. The proposed approach then converts the corresponding principal components to decision trees. These trees are defined in the original attributes-space and they are functionally orthogonal to each other. These orthogonal trees are in turn used for accurate (in many cases, with improved accuracy) and redundancy-free (in the sense of an orthogonal basis set) compact representation of large ensembles.

Section 2 presents the motivation of this work. Section 3 presents a brief overview of the Fourier spectrum of decision trees. Section 4 describes the algorithms for computing the Fourier transform of a decision tree. Section 5 offers the algorithm for computing the tree from its Fourier spectrum. Section 6 discusses orthogonal decision trees. Section 7 presents experimental results using many well-known data sets. Finally, Section 8 concludes this paper.

2 MOTIVATION

This paper extends our earlier work [10], [9], [11] on the Fourier spectrum of decision trees. The main motivation behind this approach is to create an algebraic framework for the metalevel analysis of models produced by many ensemble learning, data stream mining, distributed data mining, and other related techniques. Most of the existing techniques treat the discrete model structures such as decision trees in an ensemble primarily as a black box. Only the output of the models is considered and combined in order to produce the overall output. Fourier bases offer a compact representation of a discrete structure that allows algebraic manipulation of decision trees. For example, we can literally add two different trees, produce weighted average of the trees themselves, or perform eigen-analysis of an ensemble of trees. Fourier representation of decision trees may offer something that is philosophically similar to what spectral representation of graphs [12] offers—an algebraic representation that allows deep analysis of discrete structures.

Fourier representation allows us to bring in the rich volume of well-understood techniques from Linear Algebra and Linear Systems Theory. This opens up many exciting possibilities for future research, such as quantifying the stability of an ensemble classifier, mining, and monitoring mission-critical data streams using properties of the eigenvalues of the ensemble. This paper takes some steps toward achieving these goals.

The main contributions of this paper are listed below:

1. It offers several new analytical results regarding the properties of the Fourier spectra of decision trees.
2. It presents a detailed discussion on the Tree Construction from Fourier Spectrum (TCFS) algo-

• *H. Kargupta and H. Dutta are with the Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250.*
E-mail: {hillol, hdutta1}@csee.umbc.edu.
• *B.-H. Park is with the Computer Science and Mathematics Division, Oak Ridge National Laboratory, PO Box 2008 MS6164, Oak Ridge, TN 37831-6164. E-mail:parkbh@ornl.gov.*

Manuscript received 20 Dec. 2004; revised 29 Nov. 2005; accepted 23 Jan. 2006; published online 18 May 2006.
For information on obtaining reprints of this article, please send e-mail to:

rithm for computing a decision tree from the Fourier coefficients. This includes discussion and experimental evaluation of the TCFS algorithm. New experimental results compare the performance of the trees constructed using the TCFS technique with that of the trees constructed using standard techniques such as C4.5.

3. It discusses Orthogonal Decision Trees (ODTs) in detail and offers extensive experimental results documenting the performance of ODTs on benchmarked data sets.

The following section reviews the Fourier representation of decision trees.

3 Decision Trees and the Fourier Representation

This section reviews the Fourier representation of decision tree ensembles, introduced elsewhere [13], [14]. It also presents some new analytical results.

3.1 Decision Trees as Numeric Functions

The approach developed in this paper makes use of linear algebraic representation of the trees. In order to do that, we first need to convert the tree into a numeric tree just in case the attributes are symbolic. A decision tree defined over a domain of categorical attributes can be treated as a 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 attributes take symbolic (nonnumeric) 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. Since the proposed approach of constructing orthogonal trees uses this representation as an intermediate stage and eventually the physical tree is converted back to the exact scheme for replacing the symbols (if any), it does not matter as long as it is consistent.

Once the tree is converted to a discrete numeric function, we can also apply any appropriate analytical transformation as necessary. Fourier transformation is one such interesting possibility. Fourier representation of a function is a 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 defining the domain. This section reviews the Fourier representation of decision tree ensembles, introduced elsewhere [9].

3.2 A Brief Review of Multivariate Fourier Basis

A Fourier basis set is comprised of orthogonal functions that can be used to represent any discrete function. In other words, it is a functionally complete representation. Consider the set of all ℓ-dimensional feature vectors where the ith feature can take λ_i different discrete 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

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

where $\mathbf{j}$ and $\mathbf{x}$ are vectors of length ℓ; x_m and j_m are mth 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 vector $\mathbf{j}$ is called a *partition* and the *order* of a partition $\mathbf{j}$ is the number of nonzero 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 nonzero values, then we say the partition is of order α since the corresponding Fourier basis function depends only on those α number of variables that take nonzero values in the partition $\mathbf{j}$.

A function $f : \mathbf{X}^\ell \rightarrow \Re$, that maps an ℓ-dimensional discrete domain to a real-valued range, can be represented using the Fourier basis functions: $f(\mathbf{x}) = \sum_{\mathbf{j}} w_{\mathbf{j}}\overline{\psi}_{\mathbf{j}}^{\overline{\lambda}}(\mathbf{x})$, where $w_{\mathbf{j}}$ is the Fourier Coefficient (FC) corresponding to the partition $\mathbf{j}$ and $\overline{\psi}_{\mathbf{j}}^{\overline{\lambda}}(\mathbf{x})$ is the complex conjugate of $\psi_{\mathbf{j}}^{\overline{\lambda}}(\mathbf{x})$; $w_{\mathbf{j}} = \sum_{\mathbf{x}} \psi_{\mathbf{j}}^{\overline{\lambda}}(\mathbf{x})f(\mathbf{x})$. The Fourier coefficient $w_{\mathbf{j}}$ can be viewed as the relative contribution of the partition $\mathbf{j}$ to the function value of $f(\mathbf{x})$. Therefore, the absolute value of $w_{\mathbf{j}}$ can be used as the "significance" of the corresponding partition $\mathbf{j}$. If the magnitude of some $w_{\mathbf{j}}$ is very small compared to other coefficients, we may consider the $\mathbf{j}$th partition to be insignificant and neglect its contribution. The *order* of a Fourier coefficient is nothing but the order of the corresponding partition. We shall often use terms like *high order* or *low order* coefficients to refer to a set of Fourier coefficients whose orders are relatively large or small, respectively. The energy of a spectrum is defined by the summation $\sum_{\mathbf{j}} w_{\mathbf{j}}^2$. Let us also define the inner product between two spectra $\mathbf{w}_{(1)}$ and $\mathbf{w}_{(2)}$, where $\mathbf{w}_{(i)} = [w_{(i),1}w_{(i),2}, \cdots w_{(i),|J|}]^T$ is the column matrix of all Fourier coefficients in an arbitrary but fixed order. Superscript T denotes the transpose operation and $|J|$ denotes the total number of coefficients in the spectrum. The inner product, $<\mathbf{w}_{(1)}, \mathbf{w}_{(2)}> = \sum_{\mathbf{j}} w_{(1),\mathbf{j}}w_{(2),\mathbf{j}}$. We will also use the definition of the inner product between a pair of real-valued functions defined over some domain Ω. This is defined as $< f_1(\mathbf{x}), f_2(\mathbf{x}) > = \sum_{\mathbf{x}\in\Omega} f_1(\mathbf{x})f_2(\mathbf{x})$.

The following section considers the Fourier spectrum of decision trees and discusses some of its useful properties.

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. [15] noted that the Fourier spectrum of an AC^0 circuit has very interesting properties and proved the following lemma:

Lemma 1 ([15]). *Let M and d be the size and depth of an AC^0 circuit. Then,*

$$\sum_{\{\mathbf{j}|o(\mathbf{j})>t\}} w_{\mathbf{j}}^2 \leq 2M2^{-t^{1/d}/20},$$

where $o(\mathbf{j})$ denotes the order (the number of nonzero variable) of partition $\mathbf{j}$ and t is a nonnegative 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:

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 the Fourier spectrum also holds for non-Boolean decision trees. The complete proof is given in the Appendix, which can be found on the Computer Society Digital Library at http://computer.org.tkde/archives.htm.

There are two additional important characteristics of the Fourier spectrum of a decision tree that we will use in this paper:

1. The Fourier spectrum of a decision tree can be efficiently computed [9].
2. The Fourier spectrum can be directly used for constructing the tree.

In other words, we can go back and forth between the tree and its spectrum. This is philosophically similar to the switching between the time and frequency domains in the traditional application of Fourier analysis for signal processing. These two issues will be discussed in details later in this paper. However, before that, we would like to make a note of one additional property.

The Fourier transformation of decision trees preserves the inner product. The functional behavior of a decision tree is defined by the class labels it assigns. Therefore, if $\{\mathbf{x}_1, \mathbf{x}_2, \cdots \mathbf{x}_{|\Omega|}\}$ are the members of the domain Ω, then the functional behavior of a decision tree $f(\mathbf{x})$ can be captured by the vector $[f]_{x\in\Omega} = [f(\mathbf{x}_1)f(\mathbf{x}_2)\cdots f(\mathbf{x}_{|\Omega|})]^T$, where the superscript T denotes the transpose operation. The following lemma proves that the inner product between two such vectors is identical to the same in between their respective Fourier spectra:

Lemma 2. *Given two functions* $f_1(\mathbf{x}) = \sum_{\mathbf{j}} w_{(1),\mathbf{j}}\overline{\psi_{\mathbf{j}}}^{\lambda}(\mathbf{x})$ *and*

$f_2(\mathbf{x}) = \sum_{\mathbf{j}} w_{(2),\mathbf{j}}\overline{\psi_{\mathbf{j}}}^{\lambda}(\mathbf{x})$ *in Fourier representation, then,*

$< f_1(\mathbf{x}), f_2(\mathbf{x}) > = < \mathbf{w}_{(1)}, \mathbf{w}_{(2)} > .$

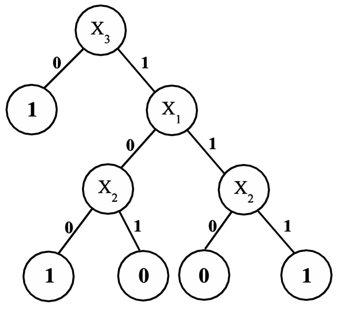


Fig. 1. A Boolean decision tree.

Proof.

$$\begin{aligned}< f_1(\mathbf{x}), f_2(\mathbf{x}) > &= \sum_{\mathbf{x}\in\Omega} f_1(\mathbf{x})f_2(\mathbf{x})\\ &= \sum_{\mathbf{x}\in\Omega}\sum_{\mathbf{j},\mathbf{i}} w_{(1),\mathbf{j}}\overline{\psi_{\mathbf{j}}}^{\lambda}(\mathbf{x})w_{(2),\mathbf{i}}\overline{\psi_{\mathbf{i}}}^{\lambda}(\mathbf{x})\\ &= \sum_{\mathbf{j},\mathbf{i}} w_{(1),\mathbf{j}}w_{(2),\mathbf{i}}\sum_{\mathbf{x}\in\Omega}\overline{\psi_{\mathbf{j}}}^{\lambda}(\mathbf{x})\overline{\psi_{\mathbf{i}}}^{\lambda}(\mathbf{x})\\ &= \sum_{\mathbf{j}} w_{(1),\mathbf{j}}w_{(2),\mathbf{j}}\\ &=< \mathbf{w}_{(1)}, \mathbf{w}_{(2)} > .\end{aligned}$$

□

The fourth step is true since Fourier basis functions are orthonormal.

4 Computing the Fourier Transform of a Decision Tree

The Fourier spectrum of a given tree can be computed efficiently by 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 Fig. 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 ith 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 $\{(x_3 \overset{1}{\rightarrow} x_1, x_1 \overset{0}{\rightarrow} x_2\}$ in Fig. 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 shall use the term *order* to represent the number of nonwildcard values in a schema. The following section describes an 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:

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

where Λ denotes the complete instance space, S_{l_i} is an instance subspace which the ith 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 3. *For any Fourier basis function* $\psi_{\mathbf{j}}$, $\sum_{\mathbf{x}\in\Lambda} \psi_j(x) = 0$.

Proof. Since Fourier basis functions form an orthogonal set,

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

Here, ψ_0 is the 0th Fourier basis function, which is constant (one) for all $\mathbf{x}$. □

Lemma 4. *Let* $\mathbf{h_i}$ *be a schema defined by the path to a leaf node* l_i. *Then, if* $\mathbf{j}$ *has a nonzero attribute value at a position where* $\mathbf{h_i}$ *has no value (wild-card),*

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

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 $\mathbf{h_i}$ 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,

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

□

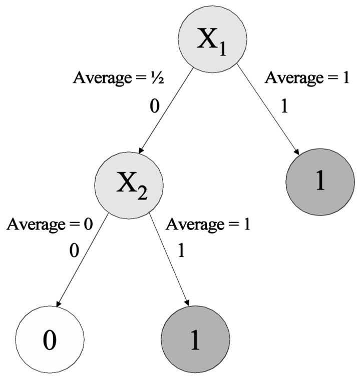


Fig. 2. An instance of a Boolean decision tree that shows average output values at each subtree.

Lemma 5. *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 4. □

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

Before describing the algorithm, we need to introduce some notations. Let $\mathbf{h}_{k=i}$ be a vector that is generated by replacing the kth position of $\mathbf{h}$ with value i. Note that this notation will be used for both schema and partition. Let us consider a nonleaf 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 function value of domain members covered by a subtree accessible through the ith child of n. For example, in Fig. 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 the ith subtree of the node where x_k appears.

The algorithm starts with precalculating all $F_{x_k}(i)$s (This is essentially a recursive "Tree-Visit" operation). Then, it incrementally finds nonzero FCs as it traverses the tree. If we let $\mathbf{S}$ denote the set of partitions that correspond to nonzero FCs, initially, $\mathbf{S} = \{000\ldots0\}$ and the corresponding $w_{000\ldots0}$ is calculated with the overall average of output. In Fig. 2, it is: $\frac{1}{2} \times \frac{1}{4} + \frac{1}{2} \times 1 = \frac{5}{8}$. The algorithm continues to extract all remaining nonzero FCs in recursive fashion from the root. New nonzero FCs are identified by inducing their correponding partitions from the existing $\mathbf{S}$. For any $\mathbf{h} \in S$, when a node with the feature x_k is visited, partitions $\mathbf{h}_{k=1}, \cdots, \mathbf{h}_{k=\lambda_k-1}$ are added into $\mathbf{S}$, where λ_k is the cardinality of x_k. For the tree in Fig. 2, $\mathbf{S}$ is initially $\{000\}$. Then, 010 is added to $\mathbf{S}$ when x_1 is visited. Note that 010 is found by replacing the first position (starting from zero) with 1, i.e., $h_{1=1} = 010$ is obtained from $\mathbf{h} = 000$. w_{010} is computed using (1):

```
Function ExtractFS(input: Partition Set $\mathbf{S}$,
Node $node$, Schema $\mathbf{h}$)
  $x_k \leftarrow$ feature appearing in $node$
  $\mathbf{N} \leftarrow \phi$
  for each $\mathbf{j} \in \mathbf{S}$
    for each $i$ from $(1, \cdots, \lambda_k - 1)$
      $\mathbf{N} \leftarrow \mathbf{N} \cup \{\mathbf{j}_{k=i}\}$
    end
  end
  $size \leftarrow \frac{|node|}{\lambda_k |\Lambda|}$
  for each $\mathbf{j} \in \mathbf{N}$
    for each $i$ from $(0, \cdots, \lambda_k - 1)$
      $w_{\mathbf{j}} \leftarrow w_{\mathbf{j}} + size \times F_{x_k}(i)\psi_{\mathbf{j}}(\mathbf{h}_{k=i})$
    end
  end
  $\mathbf{S} \leftarrow \mathbf{S} \cup \mathbf{N}$
  for each $i$ from $(0, \cdots, \lambda_k - 1)$
    ExtractFS($\mathbf{S}$, $node_i$, $\mathbf{h}_{k=i}$)
  end
end
```

Fig. 3. Algorithm for obtaining the Fourier spectrum of a decision tree. k in x_k implies that x_k is the kth feature. λ_k denotes the cardinality of x_k and $|node|$ denotes the size of subspace $node$ covers. $|\Lambda|$ is the size of the complete instance space. $node_i$ is the ith child of $node$.

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

For x_2, $\{001, 011\}$ will be added into $\mathbf{S}$. w_{001} and w_{011} are computed similarly as w_{010}. The pseudocode of the algorithm is presented in Fig. 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.

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

where $f_i(\mathbf{x})$ and a_i are ith decision tree and its weight, respectively. J_i is set of nonzero Fourier coefficients that are detected by the ith decision tree and $w_{\mathbf{j}}^{(i)}$ is a Fourier coefficient in J_i. Now, (2) 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 Multiclass Decision Trees

A multiclass 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 a 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 multiclass 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 ith class. In other words, we treat the tree to have a Boolean classification with respect to the ith 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: $f(\mathbf{x}) = c_1 f^{(1)}(\mathbf{x}) + c_2 f^{(2)}(\mathbf{x}) + \cdots + c_l f^{(l)}(\mathbf{x})$, where each c_i corresponds to a mapped value for the ith classification. Note that if $\mathbf{x}$ belongs to jth 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 ith tree in the ensemble and its partial function for the kth classification, respectively. Finally, the classification of an ensemble, of a decision tree that adopts voting as its aggregation scheme can be defined as: $f(x) = \mathrm{argmax}_k(f^{(k)}(x))$.

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 that the exponential decay property is also true for a Fourier spectrum of non-Boolean decision trees. In the following 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

Consider a classification problem with Boolean class labels—$\{0, 1\}$. Recall that a schema $\mathbf{h}$ denotes a path to a node n_k in a decision tree. In order to compute the information gain introduced by splitting the node using a particular attribute, we first need to compute the entropy of the class distribution at that node. We do that by introducing a quantity called *schema average*. Let us define the *schema average* function value as follows:

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

where $f(\mathbf{x})$ is the classification value of $\mathbf{x}$ and $|\mathbf{h}|$ denotes the number of members in schema $\mathbf{h}$. Note that the schema average $\phi(\mathbf{h})$ is nothing but the frequency of all instances of the schema $\mathbf{h}$ with a classification value of 1. Similarly, note that the frequency of the tuples with classification value of 0 is $(1 - \phi(\mathbf{h}))$. It can therefore be used to compute the entropy at the node n_k.

$$\text{confidence}(\mathbf{h}) = \max(\phi(\mathbf{h}), 1 - \phi(\mathbf{h}))$$
$$\text{entropy}(\mathbf{h}) = -\phi(\mathbf{h}) \log \phi(\mathbf{h}) - (1 - \phi(\mathbf{h})) \log(1 - \phi(\mathbf{h})).$$

The computation of $\phi(\mathbf{h})$ using the above expression 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:

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

where $\mathbf{h} = ***b_1***b_2***b_m***$ that has m nonwild-card values b_i at position j_i and $p_i \in \{0, 1, \ldots, \lambda_{j_i} - 1\}$. A similar Walsh analysis-based approach for analyzing the behavior of genetic algorithms can be found elsewhere [17]. Note that the summations in (3) are defined only for the fixed (nonwild-card) positions that correspond to the features defining the path to the node n_k.

Using (3) as a tool to obtain information gain, it is relatively straight-forward to come up with a version of ID3 or C4.5-like algorithms that work using the Fourier spectrum. However, a naive approach may be computationally inefficient. The computation of $\phi(\mathbf{h})$ requires an exponential number of FCs with respect to the order of $\mathbf{h}$. Thus, the cost involved in computing $\phi(\mathbf{h})$ increases exponentially as the tree becomes deeper. Moreover, since the Fourier spectrum of the ensemble is very compact in size, most Fourier coefficients involved in computing $\phi(\mathbf{h})$ are zero. Therefore, the evaluation of $\phi(\mathbf{h})$ using (3) 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 (3). When computing the average of an 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 precalculated. 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 whose orders are just one more than that of this schema. In the following section, we describe a tree construction algorithm that is based on these observations.

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 kth feature is set to i (Note that we use similar notation for ExtractFS). For example, consider schemata $\mathbf{h} = (\,*1**2)$ and $h_{3=1} = (*1*12)$. Here, we use an integer number-based indexing of the features (zero for the leftmost feature). $\pi(\mathbf{h})$ denotes a set of partitions that are required to compute $\phi(\mathbf{h})$ (See (3)). A k-fixed *partition* is a partition with a nonzero value at the kth position. Let $\xi(k)$ be a set of order one k-fixed partitions; $\gamma(h_{k=i})$ be the partial sum of $\phi(\mathbf{h}_{k=i})$ which only includes k-fixed partitions. Now, the information gain achieved by choosing the kth feature with a given $\mathbf{h}$ is redefined using these new notations:

```
1  Function TCFS(input: Fourier Spectrum FS)
2    Initialize Candidate Feature Set CFSET
3    create root node
4    h ← (***...***)
5    root ← Build(h, FS,SFSET)
6    return root
7  end
```

Fig. 4. Algorithm for constructing a decision tree from the Fourier spectrum (TCFS).

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

where $\otimes$ is the Cartesian product and λ_k is the cardinality of the kth 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 choosing the kth 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 kth 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 kth 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 to. The detection of all appropriate FCs requires a scan over the FS. However, they are removed 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 predetermined bound. A pictorial description of the algorithm is shown in Fig. 6. Pseudocode of the algorithm is presented in Figs. 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 the

```
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 $k$
    $\mathbf{h_{k=i}} \leftarrow$ 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
```

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

Fourier spectrum. The following section presents an extension of the TCFS for handling non-Boolean class labels.

5.3 Extension of TCFS to Multiclass Decision Trees

The extension of the TCFS algorithm to multiclass 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 occurrence of the ith class label in $\mathbf{h}$. Then, the entropy of a schema is redefined as follows:

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

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 the 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 orthogonal decision trees that can be constructed using the Fourier spectrum of the trees in an ensemble.

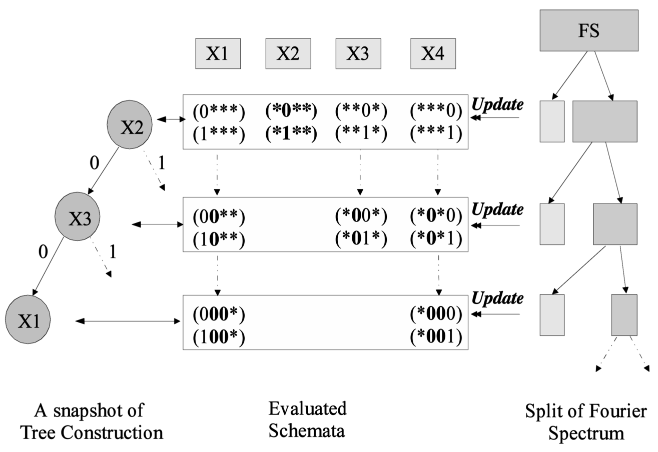


Fig. 6. Illustration of the Tree Construction from Fourier Spectrum (TCFS) algorithm. It shows the constructed tree on the left. The schemata evaluated at different orders are shown in the middle. The rightmost tree shows the splitting of the set of all Fourier coefficients used for making the process of looking up the appropriate coefficients efficient.

6 Removing Redundancies from Ensembles

Existing ensemble-learning techniques work by combining (usually a linear combination) the output of the base classifiers. They do not structurally combine the classifiers themselves. As a result, they often share a lot of redundancies. The Fourier representation offers a unique way to fundamentally aggregate the trees and perform further analysis to construct an efficient representation.

Let $f_e(\mathbf{x})$ be the underlying function representing the ensemble of m different decision trees, where the output is a weighted linear combination of the outputs of the base classifiers. Then, we can write,

$$\begin{aligned} f_e(\mathbf{x}) &= \alpha_1 \tau_{(1)}(\mathbf{x}) + \alpha_2 \tau_{(2)}(\mathbf{x}) + \cdots + \alpha_m \tau_{(m)}(\mathbf{x}) \\ &= \alpha_1 \sum_{\mathbf{j} \in \mathcal{J}_1} w_{(1),\mathbf{j}} \overline{\psi}_{\mathbf{j}}^{\lambda}(\mathbf{x}) + \cdots + \alpha_m \sum_{\mathbf{j} \in \mathcal{J}_m} w_{(m),\mathbf{j}} \overline{\psi}_{\mathbf{j}}^{\lambda}(\mathbf{x}), \end{aligned}$$

where α_i is the weight of the ith decision tree and $\mathcal{J}_i$ is the set of all partitions with nonzero Fourier coefficients in its spectrum. Therefore, $f_e(\mathbf{x}) = \sum_{\mathbf{j} \in \mathcal{J}} w_{(e),\mathbf{j}} \overline{\psi}_{\mathbf{j}}^{\lambda}(\mathbf{x})$, where $w_{(e),\mathbf{j}} = \sum_{i=1}^{m} \alpha_i w_{(i),\mathbf{j}}$ and $\mathcal{J} = \cup_{i=1}^{m} \mathcal{J}_i$. Therefore, the Fourier spectrum of $f_e(\mathbf{x})$ (a linear ensemble classifier) is simply the weighted sum of the spectra of the member trees.

Consider the matrix D where $D_{i,j} = \tau_{(j)}(\mathbf{x_i})$, where $\tau_{(j)}(\mathbf{x_i})$ is the output of the tree $\tau_{(j)}$ for input $\mathbf{x_i} \in \Omega$. D is an $|\Omega| \times m$ matrix where $|\Omega|$ is the size of the input domain and m is the total number of trees in the ensemble.

An ensemble classifier that combines the outputs of the base classifiers can be viewed as a function defined over the set of all rows in D. If $D_{*,j}$ denotes the jth column matrix of D, then the ensemble classifier can be viewed as a function of $D_{*,1}, D_{*,2}, \cdots D_{*,m}$. When the ensemble classifier is a linear combination of the outputs of the base classifiers, we have $F = \alpha_1 D_{*,1} + \alpha_2 D_{*,2} + \cdots \alpha_m D_{*,m}$, where F is the column matrix of the overall ensemble-output. Since the base classifiers may have redundancy, we would like to construct a compact low-dimensional representation of the matrix D. However, explicit construction and manipulation of the matrix D is difficult, since most practical applications deal with a very large domain. We can try to construct an approximation of D using only the available training data. One such approximation of D and its Principal Component Analysis-based projection is reported elsewhere [18]. Their technique performs PCA of the matrix D, projects the data in the representation defined by the eigenvectors of the covariance matrix of D, and then performs linear regression for computing the coefficients $\alpha_1, \alpha_2, \cdots,$ and α_m.

While the approach is interesting, it has a serious limitation. First of all, the construction of an approximation of D even for the training data is computationally prohibiting for most large-scale data mining applications. Moreover, this is an approximation since the matrix is computed only over the observed data set of the entire domain. In the following, we demonstrate a novel way to perform a PCA of the matrix containing the Fourier spectra of trees. The approach works without explicitly generating the matrix D. It is important to note that the PCA-based regression scheme [18] offers a way to find the weightage for the members of the ensemble. It does not offer any way to aggregate the tree structures and construct a new representation of the ensemble which the current approach does.

The following analysis will assume that the columns of the matrix D are mean-zero. This restriction can be easily removed with a simple extension of the analysis. Note that the covariance of the matrix D is $D^T D$. Let us denote this covariance matrix by C. The (i, j)th entry of the matrix,

$$\begin{aligned} C_{i,j} &= < D(*, i), D(*, j) > = < \tau_{(i)}(\mathbf{x}), \tau_{(j)}(\mathbf{x}) > \\ &= \sum_{\mathbf{p}} w_{(i),\mathbf{p}} w_{(j),\mathbf{p}} = < \mathbf{w}_{(i)}, \mathbf{w}_{(j)} > . \end{aligned} \tag{4}$$

The fourth step is true by Lemma 2. Now, let us consider the matrix W, where $W_{i,j}$ is the coefficient corresponding to the ith member of the partition set $\mathcal{J}$ from the spectrum of the tree $\tau_{(j)}$. Equation (4) implies that the covariance matrices of D and W are identical. Note that W is an $|\mathcal{J}| \times m$ dimensional matrix. For most practical applications, $|\mathcal{J}| << |\Omega|$. Therefore, analyzing W using techniques like PCA is significantly easier. The following discourse outlines a PCA-based approach.

PCA of the covariance matrix of W produces a set of eigenvectors $V_1, V_2, \cdots V_k$. The eigenvalue decomposition constructs a new representation of the underlying domain. Note that since the eigenvectors are nothing but a linear combination of the original column vectors of W, each of them also form a Fourier spectrum and we can reconstruct a decision tree from this spectrum. Moreover, since they are orthogonal to each other, the tree constructed from them also maintain, the orthogonality condition and, therefore, they are redundancy-free. They define a basis set and can be used to represent any given decision tree in the ensemble in the form of a linear combination. Orthogonal decision trees can be defined as an immediate extension of this framework.

A pair of decision trees $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ are orthogonal to each other if and only if $< f_a(\mathbf{x}), f_b(\mathbf{x}) >= 0$ when $a \neq b$ and $< f_a(\mathbf{x}), f_b(\mathbf{x}) >= 1$ otherwise. The second condition is actually a slightly special case of orthogonal functions —orthonormal condition. A set of trees are pairwise orthogonal if every possible pair of members of this set satisfy the orthogonality condition.

The orthogonality condition guarantees that the representation is not redundant. These orthogonal trees form a basis set that spans the entire function space of the ensemble. The overall output of the ensemble is computed from the output of these orthogonal trees. Specific details of the ensemble output computation depends on the adopted technique to compute the overall output of the original ensemble. However, for most popular cases considered here, it boils down to computing the average output. If we choose to go for weighted average, we may need to compute the coefficients corresponding to each V_q using linear regression or other similar techniques.

7 Experimental Results

This section reports the experimental performance of orthogonal decision trees on the following data sets—SPECT, NASDAQ, DNA, House of Votes, and Contraceptive Method Usage Data. For each data set, the following three experiments are performed using known classification techniques:

1. **C4.5**: The C4.5 classifier is built on training data and validated over test data.
2. **Bagging**: A popular ensemble classification technique, bagging, is used to test the classification accuracy of the data set.
3. **Random Forest**: Random forests are built on the training data, using approximately half the number of features in the original data set. The number of

trees in the forest is identical to that used in the bagging experiment.[1]

We then perform another set of experiments for comparing the techniques described in the previous sections in terms of error in classification and tree complexity.

1. **Reconstructed Fourier Tree (RFT)**: The training set is uniformly sampled, with replacement and C4.5 trees built on each sample. The Fourier representation of each individual tree is obtained, preserving a certain percentage (e.g., 90 percent) of the energy. This representation of a tree is used to reconstruct a decision tree using the TCFS algorithm described in Section 5. The performance of a reconstructed Fourier tree is compared with the original C4.5 tree. The error in classification and tree complexity of each of the reconstructed trees is reported. The purpose of this experiment is to study the effect of "noise removal" from the ensemble on its classification-accuracy by going to the Fourier domain and then coming back to the tree domain using the TCFS algorithm.
2. **Aggregated Fourier Tree (AFT)**: The training set is uniformly sampled, with replacement and C4.5 decision trees built on each sample (This is identical to bagging). A Fourier representation of each tree is obtained (preserving a certain percentage of the total energy), and these are aggregated with uniform weighting to obtain the spectrum of an Aggregated Fourier Tree (AFT). The AFT is reconstructed using the TCFS algorithm described before and the classification accuracy and the tree complexity of this aggregated Fourier tree is reported.
3. **Orthogonal Decision Trees**: The matrix containing the Fourier coefficients of the decision trees is subjected to principal component analysis. Orthogonal trees are built using the corresponding eigenvectors. In most cases, it is found that the first principal eigenvector captures most of the variance and, thus, the orthogonal decision tree constructed from this eigenvector is of particular interest. We report the error in classification and tree complexity of the orthogonal decision tree obtained from the most dominant eigenvector. We also perform experiments where we keep k[2] significant eigenvectors. The trees are combined by weighting them according to the coefficients obtained from a Least Square Regression. Each orthogonal decision tree is weighted using coefficients calculated from Least Square Regression. For this, we allow all the orthogonal decision trees to individually produce their classification on the test set. Thus, each ODT produces a column vector of its classification estimate. Since the class-labels in the test set are already known, we use the least square regression to obtain the weights to assign to each ODT. The accuracy of the orthogonal decision trees is reported as ODT-LR (ODTs combined using Least Square Regression).

In addition to reporting the error in classification, we also report the tree complexity, the total number of nodes in the tree. Similarly, the term ensemble complexity reflects the total number of nodes in all the trees in the ensemble. A smaller ensemble tree complexity implies a compact representation of an ensemble and, therefore, it is desirable. Our experiments show that ODTs usually offer significantly reduced ensemble tree complexity without any reduction in the accuracy. The following section presents the results for the SPECT data set.

7.1 SPECT Data Set

This section illustrates the idea of orthogonal decision trees using a well-known binary data set. The data set, available from the University of California at Irvine, Machine Learning Repository, describes the diagnosing of cardiac Single Proton Emission Computed Tomography (SPECT) images into two categories, normal or abnormal. The database of 267 SPECT image sets (patients) is processed to extract features that summarize the original SPECT images. As a result, 44 continuous feature patterns are obtained for each patient, which are further processed to obtain 22 binary feature patterns. The training data set consists of 80 instances and 22 attributes. All the features are binary, and the class label is also binary (depending on whether a patient is deemed normal or abnormal). The test data set consists of 187 instances and 22 attributes.

Table 1a shows the error percentage obtained in each of the different classification schemes. The root mean squared error for the 10-fold cross validation in the C4.5 experiment is found to be 0.4803 and the standard deviation is 2.3862. For Bagging, the number of trees in the ensemble is chosen to be 40. Our experiments reveal that further increase in number of trees in the ensemble causes a decrease in accuracy of classification of the ensemble possibly due to overfitting of the data.

For experiments with Random Forests, a forest of 40 trees, each constructed while considering 12 random features, is built. The average Out of bag error is reported to be 0.3245.

Fig. 7a compares the accuracy of the original C4.5 ensemble with that of the Reconstructed Fourier Tree (RFT) ensemble preserving 90 percent of the energy of the spectrum. The results reveal that if all of the spectrum is preserved, the accuracy of the original C4.5 tree and RFT are identical. When the higher order Fourier coefficients are removed, this becomes equivalent to pruning a decision tree. This explains the higher accuracy of the reconstructed Fourier tree preserving 90 percent of the energy of the spectrum. Fig. 7b compares the tree complexity of the original C4.5 ensemble with that of the RFT ensemble.

In order to construct the orthogonal decision trees, the coefficient matrix is projected onto the first 15 most significant principal components. The most significant principal component captures 85.1048 percent of the variance and the tree complexity of the ODT constructed from this component is 17 with an accuracy of 91.97 percent. Fig. 8 shows the variance captured by all the 15 principal components.

Tables 1a and 1b present the accuracy and the tree-complexity for this data set, respectively. The orthogonal trees are found to be smaller in complexity, thus reducing the complexity of the ensemble.

7.2 NASDAQ Data Set

The NASDAQ data set is a semisynthetic data set with 1,000 instances and 100 discrete attributes. The original data set has three years of NASDAQ stock quote data. It is

1. We used the WEKA implementation (http://www.cs.waikato.ac.nz/ml/weka/) of Bagging and Random Forests.

2. We select the value of k in such a manner that the total variance captured is more than 90 percent. One could potentially do cross-validation to obtain a suitable value of k as pointed out in [19], but this is beyond the current scope of the work and will be explored in future.

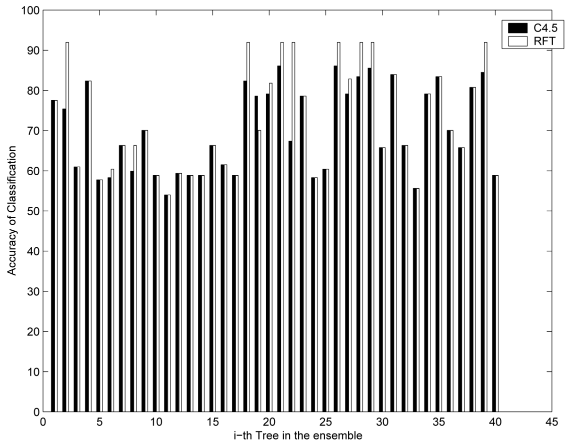
TABLE 1
(a) Classification Error and (b) Tree Complexity for SPECT Data

Method of classification	Error Percentage
C4.5	24.5989 (%)
Bagging	20.85 (%)
Random Forest	22.99466 (%)
Aggregated Fourier Tree (AFT)	19.78(%)
ODT from 1st PC	8.02(%)
ODT-LR	8.02(%)

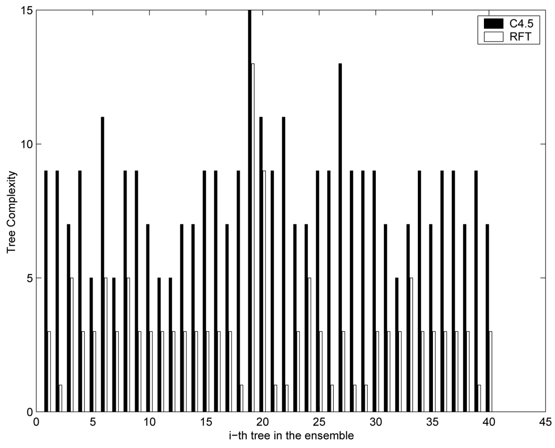
(a)

Method of classification	Tree Complexity
C4.5	13
Bagging (average of 40 trees)	5.06
Random Forest (average of 40 trees)	49.67
Aggregated Fourier Tree(AFT)(40 trees)	3
Orthogonal Decision Tree from 1st PC	17
Orthogonal Decision Trees (average of 15 trees)	4.3

(b)



(a)



(b)

Fig. 7. The accuracy and tree complexity of C4.5 and RFT for SPECT data.

preprocessed and transformed to discrete data by encoding percentages of changes in stock quotes between consecutive days. For these experiments, we assign four discrete values, that denote levels of changes. The class labels predict whether the Yahoo stock is likely to increase or decrease based on attribute values of the 99 stocks. We randomly select 200 instances for training and the remaining 800 instances forms the test data set.

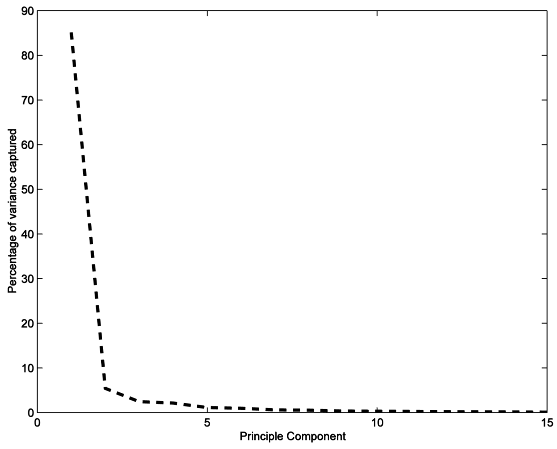


Fig. 8. Percentage of variance captured by principal components for SPECT Data.

Table 2a illustrates the classification accuracies of different experiments performed on this data set. The root mean squared error for the 10-fold cross validation in the C4.5 experiment is found to be 0.4818 and the standard deviation is 2.2247. C4.5 has the best classification accuracy, though the tree built has the highest tree complexity also. For the bagging experiment, C4.5 trees are built on the data set, such that the size of each bag (used to build the tree) as a percentage of the data set is 40 percent. Also, a Random Forest of 60 trees, each constructed while considering 50 random features, is built on the training data and tested with the test data set. The average out of bag error is reported to be 0.3165.

Fig. 9a compares the accuracy of the original C4.5 ensemble with that of the Reconstructed Fourier Tree (RFT) ensemble preserving 90 percent of the energy of the spectrum. Fig. 9b compares the tree complexity of the original C4.5 ensemble with that of the RFT ensemble.

For the orthogonal trees, we project the data along the first 10 most significant principal components. Fig. 10 illustrates the percentage of variance captured by the 10 most significant principal components.

Table 2b presents the tree-complexity information for this set of experiments. Both the aggregated Fourier tree and the orthogonal trees performed better than the single

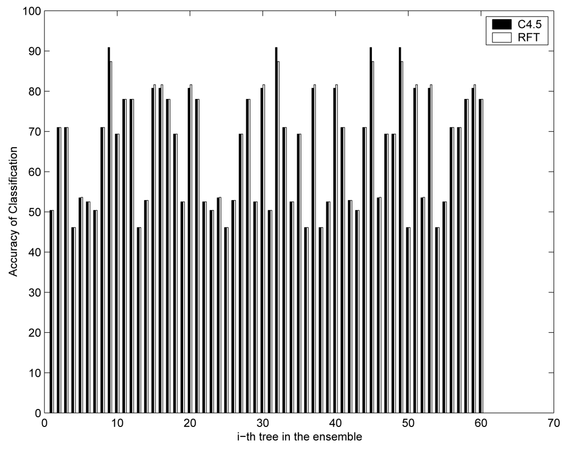
TABLE 2
(a) Classification Error and (b) Tree Complexity for NASDAQ Data

Method of classification	Error Percentage
C4.5	24.63 (%)
Bagging	32.75 (%)
Random Forest	25.75 (%)
Aggregated Fourier Tree(AFT)	34.51 (%)
ODT from 1st PC	31.12(%)
ODT-LR	31.12(%)

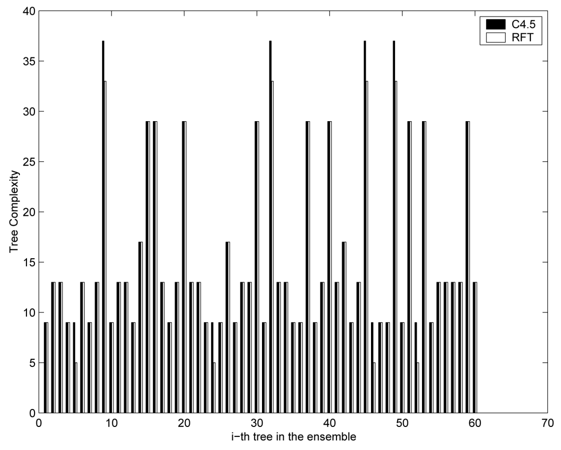
(a)

Method of classification	Tree Complexity
C4.5	29
Bagging (average of 60 trees)	17
Random Forest (average of 60 trees)	45.71
Aggregated Fourier Tree(AFT) (60 trees)	15.2
Orthogonal Decision Tree from 1st PC	3
Orthogonal Decision Trees (average of 10 trees)	6.2

(b)



(a)



(b)

Fig. 9. The accuracy and tree complexity of C4.5 and RFT for Nasdaq data.

C4.5 tree or bagging. The tree-complexity result appears to be quite interesting. While a single C4.5 tree had 29 nodes in it, the orthogonal tree from the first principal component requires just three nodes, which is clearly a much more compact representation.

7.3 DNA Data Set

The DNA data set[3] is a processed version of the corresponding data set available from the UC Irvine repository. The processed StatLog version replaces the symbolic attribute values representing the nucleotides (only A,C,T,G) by three binary indicator variables. Thus, the original 60 symbolic attributes are changed into 180 binary attributes. The nucleotides A,C,G,T are given indicator values as follows: $A = 100$, $C = 010$, $G = 001$, and $T = 000$. The data set has three class values 1, 2, and 3 corresponding to exon-intron boundaries (sometimes called **acceptors**), intron-exon boundaries (sometimes called **donors**), and the case when neither is true. We further process the data such that, there are are only two class labels, i.e., class 1 represents either donors or acceptors, while class 0 represents neither. The training set consists of 2,000 instances and 180 attributes of which 47.45 percent belongs to class 1, while the remaining 52.55 percent belongs to class 0. The test data set consists of 1,186 instances and 180 attributes of which 49.16 percent belongs to class 0 while the remaining 50.84 percent belongs to the class 1. Table 3a reports the classification error. The root mean squared error for the 10-fold cross validation in the C4.5 experiment is found to be 0.2263 and the standard deviation is 0.6086.

It may be interesting to note that the first five eigenvectors are used in this experiment. Fig. 11 shows the variance captured by these components. As before, the redundancy free trees are combined by the weights obtained from Least Square Regression. Table 3b reports the tree complexity for this data set.

Fig. 12a compares the accuracy of the original C4.5 ensemble with that of the Reconstructed Fourier Tree (RFT) ensemble preserving 90 percent of the energy of the spectrum. Fig. 12b compares the tree complexity of the original C4.5 ensemble with that of the RFT ensemble.

7.4 House of Votes Data

The 1984 United States Congressional Voting Records Database is obtained from the University of California, Machine Learning Repository. This data set includes votes for each of the US House of Representatives Congressmen on the 16 key votes identified by the CQA including water project cost sharing, adoption of budget resolution, mx-missile, immigration, etc. It has 435 instances, 16 Boolean valued attributes, and a binary class label (democrat or

3. Obtained from http://www.liacc.up.pt/ML/statlog/data sets/dna.

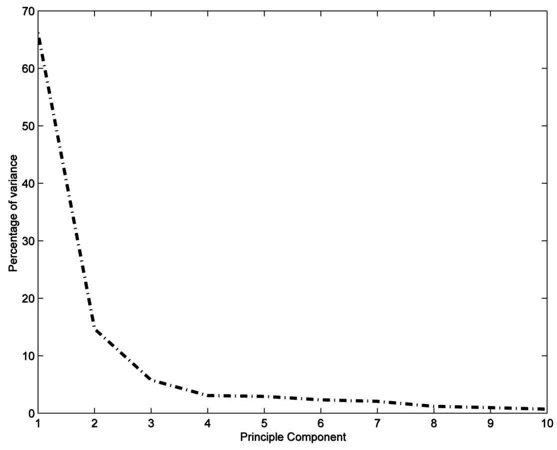


Fig. 10. Percentage of variance captured by principal components for Nasdaq Data.

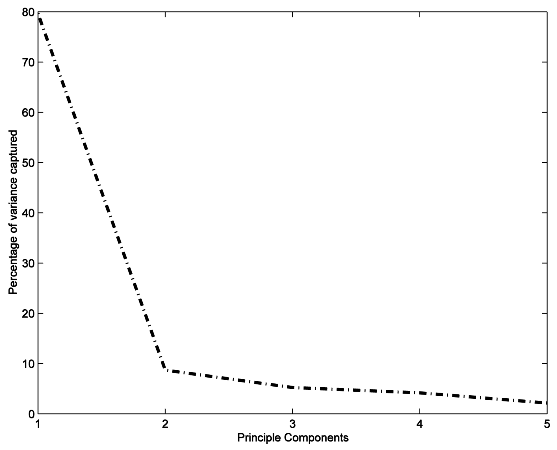


Fig. 11. Percentage of variance captured by principal components for DNA Data.

republican). Our experiments use the first 335 instances for training and the remaining 100 instances for testing. In our experiments, missing values in the data are replaced by one.

The results of classification are shown in Table 4a while the tree complexity is shown in Table 4b. The root mean squared error for the 10-fold cross validation in the C4.5 experiment is found to be 0.2634 and the standard deviation is 0.3862. For Bagging, 15 trees are constructed using the data set, since this produced the best classification results. The size of each bag was 20 percent of the training data set. Random Forest of 15 trees, each constructed by considering eight random features produces an average out of bag error of 0.05502. The accuracy of classification and the tree complexity of the original C4.5 and RFT ensemble are illustrated in Fig. 13a and Fig. 13b, respectively.

For orthogonal trees, the coefficient matrix is projected onto the first five most significant principal components. Fig. 14a illustrates the amount of variance captured by each of the principal components.

7.5 Contraceptive Method Usage Data

This data set is obtained from the University of California Irvine, Machine Learning Repository and is a subset of the 1987 National Indonesia Contraceptive Prevalence Survey. The samples were married women who were either not pregnant or did not know if they were at the time of interview. The problem is to predict the current contraceptive method choice of a woman based on her demographic and socio-economic characteristics. There are 1,473 instances and 10 attributes including a binary class label. All attributes are processed so that they are binary. Our experiments use 1,320 instances for the training set while the rest form the test data set.

The results of classification are tabulated in Table 5a while Table 5b shows the tree complexity. The root mean squared error for the 10-fold cross validation in the C4.5 experiment is found to be 0.5111 and the standard deviation is 1.8943. For a Random Forest built with 10 trees, considering five random features produces an average error in classification of about 45.88 percent and an average out of bag error of 0.42556. Fig. 15a compares the accuracy of the original C4.5 ensemble with that of the Reconstructed Fourier Tree (RFT) ensemble preserving 90 percent of the energy of the spectrum. Fig. 15b compares the tree complexity of the original C4.5 ensemble with that of the RFT ensemble.

For ODTs, the data is projected along the first ten principal components. Fig. 14b shows the amount of variance captured by each principal component. It is interesting to note that the first principal component captures only about 61.85 percent of the variance and, thus, the corresponding ODT generated from the first principal component has a relatively high tree complexity.

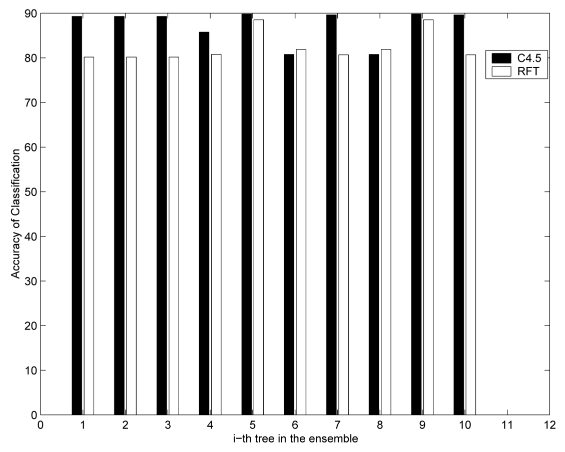
TABLE 3
(a) Classification Error and (b) Tree Complexity for DNA Data

Method of classification	Error Percentage
C4.5	6.4924 (%)
Bagging	8.9376(%)
Random Forest	4.595275 (%)
Aggregated Fourier Tree(AFT)	8.347(%)
ODT from 1st PC	10.70(%)
ODT-LR	10.70(%)

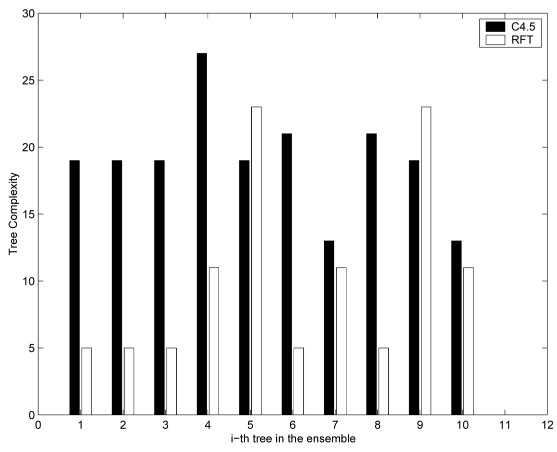
(a)

Method of classification	Tree Complexity
C4.5	131
Bagging (average of 10 trees)	34
Random Forest (average of 10 trees)	701.22
Aggregated Fourier Tree(AFT)(10 trees)	3
Orthogonal Decision Tree from 1st PC	25
Orthogonal Decision Trees (average of 5 trees)	7.4

(b)

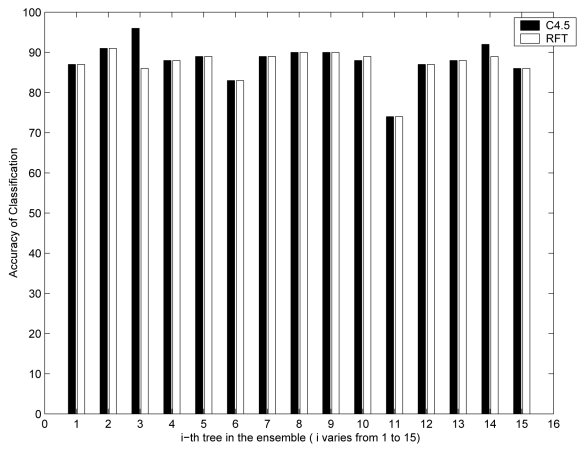


(a)

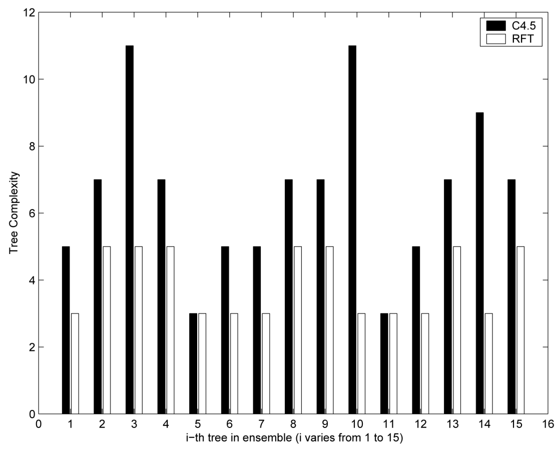


(b)

Fig. 12. The accuracy and tree complexity of C4.5 and RFT for DNA data.



(a)



(b)

Fig. 13. The accuracy and tree complexity of C4.5 and RFT for House of Votes data.

TABLE 4
(a) Classification Error and (b) Tree Complexity for House of Votes Data

Method of classification	Error Percentage
C4.5	8.0 (%)
Bagging	11.0(%)
Random Forest	5.6(%)
Aggregated Fourier Tree(AFT)	11(%)
ODT from 1st PC	11(%)
ODT-LR	11(%)

(a)

Method of classification	Tree Complexity
C4.5	9
Bagging (average of 15 trees)	5.266
Random Forest (average of 15 trees)	37.42
Aggregated Fourier Tree (AFT)(15 trees)	5
Orthogonal Decision Tree from 1st PC	5
Orthogonal Decision Trees (average of 5 trees)	3

(b)

8 Conclusions

This paper introduced the notion of orthogonal decision trees and offered a methodology to construct them. Orthogonal decision trees are functionally orthogonal to each other and they provide an efficient redundancy-free representation of large ensembles that are frequently produced by techniques like Boosting [2], [3], Bagging [4], Stacking [5], and Random Forests [6]. The proposed technique is also likely to be very useful in ensemble-based mining of distributed [10] and stream data [7], [8].

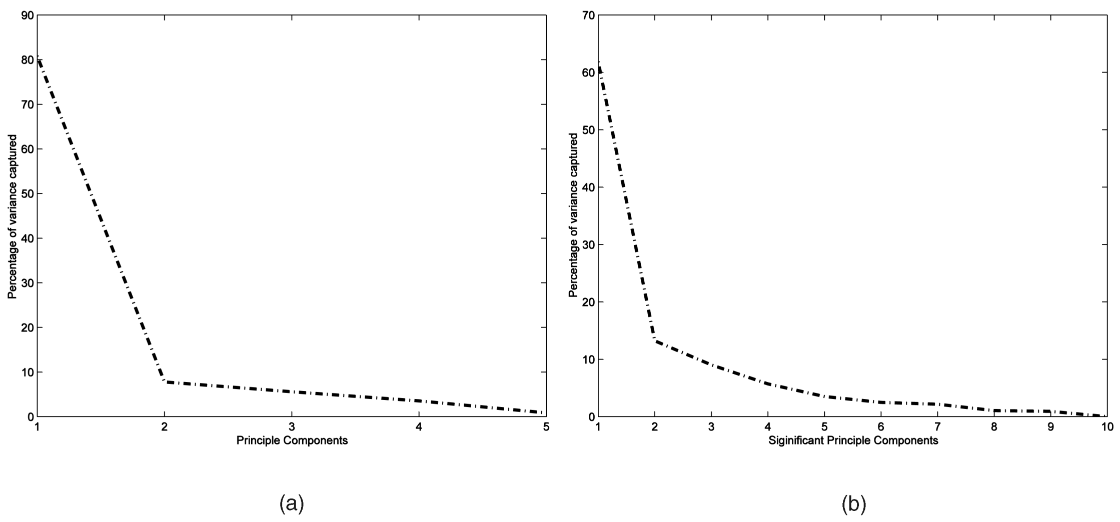


Fig. 14. Percentage of variance captured by principal components for (a) House of Votes data and (b) Contraceptive Method Usage data.

TABLE 5
(a) Classification Error and (b) Tree Complexity for Contraceptive Method Usage Data

Method of classification	Error Percentage
C4.5	49.6732(%)
Bagging	52.2876(%)
Random Forest	45.88234 (%)
Aggregated Fourier Tree(AFT)	33.98(%)
ODT from 1st PC	46.40(%)
ODT-LR	46.40(%)

(a)

Method of classification	Tree Complexity
C4.5	27
Bagging(average of 10 trees)	24.8
Random Forest (average of 10 trees)	298.11
Aggregated Fourier Tree(AFT)(10 trees)	55
Orthogonal Decision Tree from 1st PC	15
Orthogonal Decision Trees (average of 10 trees)	6.6

(b)

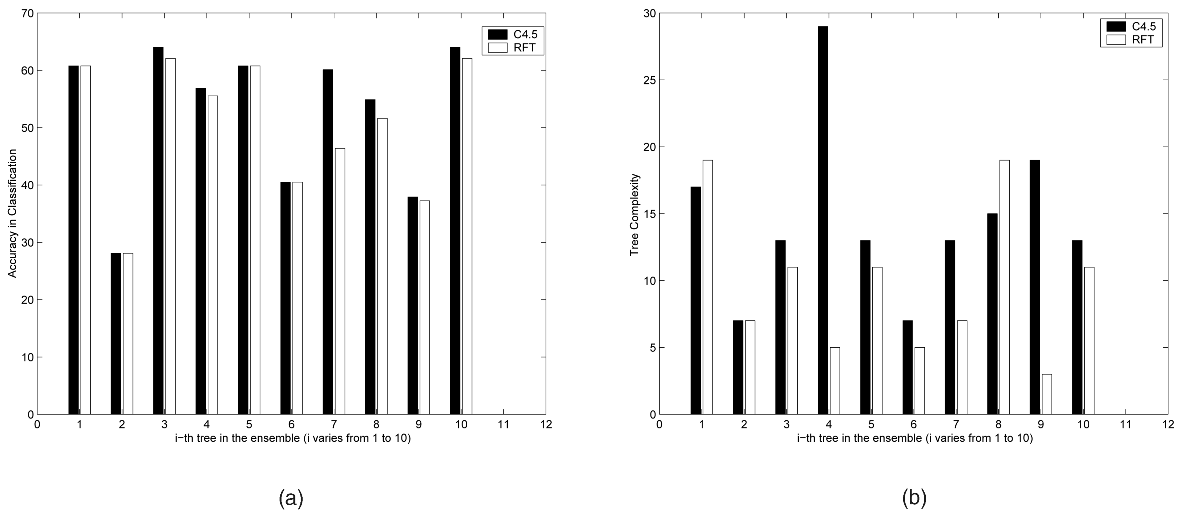


Fig. 15. The accuracy and tree complexity of C4.5 and RFT for Contraceptive Method Usage data.

The proposed approach exploits the earlier work done by Kargupta et al. [20], [9], which showed that the Fourier transform of decision trees can be efficiently computed. This work shows that we can compute the tree back from its Fourier spectrum. The paper also offered a collection of new results regarding the properties of the multivariate Fourier spectrum of decision trees. Although the paper considers the Fourier representation, this is clearly not the only available linear representation around. However, our work shows that it is particularly suitable for representing decision trees.

This work also opens up several new possibilities. Linear systems theory offers many tools for analyzing properties like stability and convergence. For example, eigenvalues of a linear system are directly associated with the stability of the system. Similar concepts may be useful in understanding the behavior of large ensembles. We plan to explore these issues in the future.

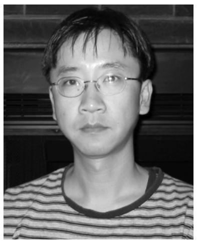
Acknowledgments

The authors acknowledge supports from the US National Science Foundation CAREER award IIS-0093353, US National Science Foundation grant IIS-0203958, and NASA grant NAS2-37143. The work of B.-H. Park was partially funded by the Scientific Data Management Center (http://sdmcenter.lbl.gov) under the Department of Energy's Scientific Discovery through Advanced Computing (DOE SciDAC) program (http://www.scidac.org). H. Kargupta is also affiliated to Agnik, LLC., Columbia, MD. A four-page version of this paper was published in the *Proceedings of the 2004 IEEE International Conference on Data Mining*.

References

[1] J.R. Quinlan, "Induction of Decision Trees," *Machine Learning,* vol. 1, no. 1, pp. 81-106, 1986.
[2] Y. Freund, "Boosting a Weak Learning Algorithm by Majority," *Information and Computation,* vol. 121, no. 2, pp. 256-285, 1995.
[3] H. Drucker and C. Cortes, "Boosting Decision Trees," *Advances in Neural Information Processing Systems,* vol. 8, pp. 479-485, 1996.
[4] L. Breiman, "Bagging Predictors," *Machine Learning,* vol. 24, no. 2, pp. 123-140, 1996.
[5] D. Wolpert, "Stacked Generalization," *Neural Networks,* vol. 5, pp. 241-259, 1992.
[6] L. Breiman, "Random Forests," *Machine Learning,* vol. 45, no. 1, pp. 5-32, 2001.
[7] W. Fan, S. Stolfo, and J. Zhang, "The Application of Adaboost for Distributed, Scalable, and On-Line Learning," *Proc. Fifth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* 1999.
[8] W.N. Street and Y. Kim, "A Streaming Ensemble Algorithm (Sea) for Large-Scale Classificaiton," *Proc. Seventh ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* 2001.
[9] H. Kargupta and B. Park, "A Fourier Spectrum-Based Approach to Represent Decision Trees for Mining Data Streams in Mobile Environments," *IEEE Trans. Knowledge and Data Eng.,* vol. 16, no. 2, pp. 216-229, 2002.
[10] B. Park, A. R, and H. Kargupta, "A Fourier Analysis-Based Approach to Learn Classifier from Distributed Heterogeneous Data," *Proc. First SIAM Int'l Conf. Data Mining,* 2001.
[11] B.H. Park and H. Kargupta, "Constructing Simpler Decision Trees from Ensemble Models Using Fourier Analysis," *Proc. Seventh Workshop Research Issues in Data Mining and Knowledge Discovery,* pp. 18-23, 2002.
[12] F. Chung, *Spectral Graph Theory.* Providence, R.I.: Am. Math. Soc., 1994.
[13] H. Kargupta and B. Park, "Mining Time-Critical Data Stream Using the Fourier Spectrum of Decision Trees," *Proc. IEEE Int'l Conf. Data Mining,* pp. 281-288, 2001.
[14] H. Kargupta, B. Park, S. Pittie, L. Liu, D. Kushraj, and K. Sarkar, "Mobimine: Monitoring the Stock Market from a PDA," *ACM SIGKDD Explorations,* vol. 3, no. 2, pp. 37-46, Jan. 2002.
[15] N. Linial, Y. Mansour, and N. Nisan, "Constant Depth Circuits, Fourier Transform, and Learnability," *J. ACM,* vol. 40, pp. 607-620, 1993.
[16] E. Kushilevitz and Y. Mansour, "Learning Decision Trees Using the Fourier Spectrum," *SIAM J. Computing,* vol. 22, no. 6, pp. 1331-1348, 1993.
[17] D. Goldberg, "Genetic Algorithms and Walsh Functions: Part I, a Gentle Introduction," *Complex Systems,* vol. 3, no. 2, pp. 129-152, 1989.
[18] C.J. Merz and M.J. Pazzani, "A Principal Components Approach to Combining Regression Estimates," *Machine Learning,* vol. 36, nos. 1-2, pp. 9-32, 1999.
[19] C. Merz and M. Pazzani, "A Principal Components Approach to Combining Regression Estimates," *Machine Learning,* vol. 36, pp. 9-32, 1999.
[20] 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,* H. Kargupta and P. Chan, eds., AAAI/MIT Press, 2000.

Hillol Kargupta received the PhD degree in computer science from the University of Illinois at Urbana-Champaign in 1996. He is an associate professor in the Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County. He is also a cofounder of Agnik LLC, an ubiquitous intelligence company. His research interests include mobile and distributed data mining and the computation in biological processes of gene expression. Dr. Kargupta won a US National Science Foundation CAREER award in 2001 for his research on ubiquitous and distributed data mining. He, along with his coauthors, received the best paper award at the 2003 IEEE International Conference on Data Mining for a paper on privacy-preserving data mining. He won the 2000 TRW Foundation Award and the 1997 Los Alamos Award for Outstanding Technical Achievement. His research has been funded by the US National Science Foundation, the US Air Force, the US Department of Homeland Security, NASA, and various other organizations. He has published more than 90 peer-reviewed articles in journals, conferences, and books. He has coedited two books: *Advances in Distributed and Parallel Knowledge Discovery* and *Data Mining: Next Generation Challenges and Future Directions*, both published by AAAI/MIT Press. He is an associate editor of the *IEEE Transactions on Knowledge and Data Engineering* and the *IEEE Transactions on Systems, Man, and Cybernetics, Part B*. He regularly serves in the organizing and program committee of many data mining conferences. More information about him can be found at http://www.cs.umbc.edu/~hillol. He is a senior member of the IEEE.



Byung-Hoon Park received the MS and PhD degrees in computer science, both from Washington State University in 1996 and 2001, respectively. He is currently a research scientist at the Computer Science and Mathematics Division of the Oak Ridge National Laboratory (ORNL). His research areas include distributed data mining, computational biology, genetic computing, data stream analysis, and text mining. His research activities have been supported by the Genomes-to-Life program of the US Department of Energy (DOE), Scientific Data Management (SDM) of DOE SciDAC program, and the Biodefense Knowledge Center projects of the US Department of Homeland Security. Before joining the ORNL, Dr. Park was with the University of Maryland Baltimore County as a postdoctoral research associate, where he was involved in a NASA EOS distributed data mining project. He served on the program committees of several data mining conferences and workshops. He also serves as a reviewer of numerous journals and conferences.

Haimonti Dutta received the BS degree in computer science from Jadavpur University, Kolkata, India, in 1999 and the MS degree in computer and information science from Temple University, Philadelphia, in 2002. She worked for a year as a Software Consultant at iGate Global Solutions, Bangalore. She is currently a PhD student in the Department of Computer Science and Electrical Engineering at the University of Maryland, Baltimore County. Her research interests include distributed data mining, data stream monitoring, grid mining, and medical informatics.