Constructing Simpler Decision Trees from Ensemble Models Using Fourier Analysis

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 is frequently used for classification and other related applications in data mining. It generates multiple models and produces the final classification by aggregating the outputs of the different models in the ensemble. However, large ensembles are often hard to interpret and difficult to translate into action-able knowledge. This paper considers the construction of a decision tree from the Fourier spectrum of an ensemble model within a user-defined range of errors. The Fourier spectrum of an ensemble of decision trees retains all the necessary information that can be used to construct a simpler "informative" decision tree. This approach can be effectively used for building ensemble-based classifiers from both static data sets and data streams.

1 Introduction

Understanding the internal structure of ensemble models is usually difficult. An ensemble model consists of multiple base models that are learned from different data subsets. It seeks to improve the prediction accuracy by aggregating its base models. However, the aggregation is conducted at the prediction level only and none of the base models are aggregated into a simpler form.

A simpler representation of an ensemble model has several merits. First, it saves storage overhead. This is particularly true when the size of the ensemble is ever increasing (e.g., in case of mining data streams). Second, a classification can be done quickly. Finally, a set of informative patterns (significant rules) can be easily extracted.

This paper considers the Fourier analysis as a tool to aggregate the trees in an ensemble, and proposes an efficient algorithm that constructs a single informative decision tree from the aggregated spectrum. Each tree in an ensemble is transformed and merged into a very compact Fourier Spectrum (FS). A tree is then constructed by computing information gain directly from the Fourier spectrum.

A large volume of research reports that ensemble classifier models often perform better than single model-based classifiers [5, 2, 10]. However, reducing an ensemble into a simpler model is a relatively new attempt and only a few have been reported. Guo and Sutiwaraphun [7] proposed knowledge probing as a technique to extract descriptive knowledge from an ensemble. They especially generated the new training data set using the ensemble to learn such knowledge. Kargupta and his colleagues considered the Fourier representation of an ensemble of decision trees in financial data stream environments [8]. Pruning meta-learning based ensemble models was suggested in [13].

The rest of the paper is organized as follows. Sections 2 introduces the Fourier transform of decision trees, and discusses several theoretical aspects of it. Section 3 describes an algorithm to construct a decision tree from the given Fourier spectrum. Section 4 discusses the empirical verification of the proposed Fourier analysis-based aggregation approach. Finally, Section 5 concludes the paper.

2 Decision Trees and the Fourier Domain

This section introduces the Fourier spectrum of a decision tree and its properties. It also shows how multiple trees can be aggregated in the Fourier domain.

2.1 A Brief Review of the Fourier Basis

The Fourier basis consists of orthogonal functions that can be used to represent any function. Consider the function space over the set of ℓ-bit discrete feature vectors. If the i-th feature x_i takes values between 0 and $n-1$, we say it has a cardinality of n. Then, the Fourier basis set that spans this space is comprised of $\Pi_{i=0}^{\ell}\lambda_i$, where λ_i represents the cardinality of the i-th feature. 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. We often call $\psi_{\mathbf{j}}^{\overline{\lambda}}(\mathbf{x})$ as the $\mathbf{j}$-th basis function.

The string $\mathbf{j}$ is called a *partition*, and the *order* of a partition $\mathbf{j}$ is the number of non-zero feature values in $\mathbf{j}$. A Fourier basis function depends on some x_i only when $j_i \neq 0$. Therefore a partition can also be viewed as a representation of a certain subset of x_i-s; every unique partition corresponds to a unique subset of x_i-s. If a partition $\mathbf{j}$ has exactly α number of 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 corresponding to the non-zeros in the partition $\mathbf{j}$.

A function $f : \mathbf{X}^{\ell} \to \Re$, that maps an ℓ-dimensional space of binary strings to a real-valued range, can be written using the Fourier basis functions: $f(\mathbf{x}) = \sum_{\mathbf{j}} w_{\mathbf{j}} \overline{\psi}_{\mathbf{j}}^{\overline{\lambda}}(\mathbf{x})$, 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 $\psi_{\mathbf{j}}^{\overline{\lambda}}(\mathbf{x})$; $w_{\mathbf{j}} = \Pi_{i=1}^{l} \frac{1}{\lambda_i} \sum_{\mathbf{x}} \psi_{\mathbf{j}}^{\overline{\lambda}}(\mathbf{x})\phi(\mathbf{x})$. 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 that of the corresponding partition. We 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.

2.2 Properties of Decision Trees in the Fourier Domain

For almost all practical purposes, decision trees have bounded depths. The following lemma illustrates interesting properties of such trees. As the proof is omitted here, interested readers may refer to [9] for Boolean case and [11] for the extension to an arbitrary discrete domain.

Lemma 1 Exponential Decay
For all $w_{\mathbf{j}}$-s in a Fourier spectrum of a decision tree,

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

where $o(\mathbf{j})$ denotes the order of partition $\mathbf{j}$, k is any non-negative integer, and $\wp(k)$ is a function that decreases exponentially in k.

The key aspect of this lemma is that the energy of higher order Fourier coefficients decays exponentially. This observation suggests that the spectrum of a decision tree (or equivalently bounded depth function) can be approximated by computing only a small number of low order Fourier coefficients. So the Fourier basis offers an efficient numeric representation of a decision tree in terms of an algebraic function that can be easily stored and manipulated. An efficient algorithm that extracts all significant Fourier coefficients from a decision tree is detailed in [12].

2.3 Fourier Spectrum of an Ensemble Classifier

The Fourier spectrum of an ensemble classifier can also be defined in the Fourier basis. Let us define $f(\mathbf{x})$ as a classification function of a ensemble classifier. In binary classification problems, $f(\mathbf{x})$ is essentially a linear weighted combination of classifications of each tree in the ensemble.

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

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 1 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 $J = \cup_{i=1}^{n} J_i$.

Therefore Fourier spectrum of $f(\mathbf{x})$ (an ensemble classifier) is an union of weighted spectrum of each tree.

3 Construction of a Decision Tree from Fourier Spectrum

This section discusses a tree construction from the Fourier spectrum using the information gain. In particular, we propose a method to compute entropies directly from the spectrum.

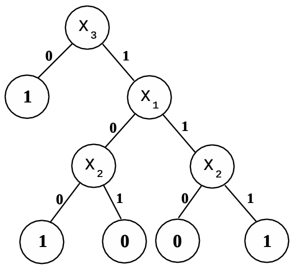

Figure 1: A Boolean decision tree

3.1 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 $f : X^\ell \to \{0,1\}$. 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 data subsets 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.

3.2 Schema Average and Information Gain

Let us consider the schema average function,

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

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

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:

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

The computation of $\phi(\mathbf{h})$ using Equation 1 and a given ensemble is not practically feasible, since we need to evaluate all $\mathbf{x} \in \mathbf{h}$. Instead, we can use the *schema transform* [6] that computes $\phi(\mathbf{h})$ directly from the given FS:

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

where $\mathbf{h}$ has m non-wildcard value b_i at position j_i and l_i has the cardinality of λ_i. Further details of the schema transform can be found in [6].

Using Equation 2 as a tool to obtain information gain, implementing a variation of C4.5 algorithm is immediately possible. 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 more deep. 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 2 is not only inefficient but also involves unnecessary computations.

An efficient algorithm that computes $\phi(\mathbf{h})$ is possible by taking advantage of a recursive and decomposable property of Equation 2. When computing the average of order l schema $\mathbf{h}$, we can reduce some computational steps if any of order l-1 schemata which subsumes $\mathbf{h}$ is already evaluated. For a simple example in a Boolean domain, let us consider the evaluation of schema $\phi(*1*0**)$. Let us also assume that $\phi(*1***)$ is pre-calculated. Then, $\phi(*1*0**)$ is obtained by simply adding w_{000100} and $-w_{010100}$ to $\phi(*1***)$. This observation leads us to an efficient algorithm to evaluate schemata. 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 which are subsumed by the current schema and whose orders are one higher. In the following section, we describe a tree construction algorithm that is based on this.

3.3 Bottom-up Approach to Construct a Tree

Before describing the algorithm, we need to introduce some notations. $\mathbf{h}_{k=i}$ is a schema whose order is one higher than that of $\mathbf{h}$ and whose non-wild card values are the same as in $\mathbf{h}$, except at the k-th feature, which is set to i. As an example, for the schema $\mathbf{h} =$ (*1**2), one such schema is $\mathbf{h}_{3=1} =$ (*1*12). Here we assign an integer number for a feature name (zero for the leftmost feature). $\pi(\mathbf{h})$ denotes a set of partitions that are required to compute $\phi(\mathbf{h})$ (See Equation 2). A k-fixed partition has a non-zero value at the k-th position. In particular, $\xi(k)$ denotes a set of order one k-fixed partitions. $\gamma(\mathbf{h}_{k=i})$ is 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:

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

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 $\mathbf{h}_{n=i}$-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 $\phi(\mathbf{h}_{n=i})$ is, in fact, computable by adding $\gamma(\mathbf{h}_{n=i})$ 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 much less time to compute higher order schemata, which goes against the naive implementation. The algorithm stops growing a path when either the original FS becomes an empty set or the minimum confidence level is met. Optionally, the depth of the resulting tree can be set as a hard bound. A pictorial description of the algorithm is shown in Figure 2.

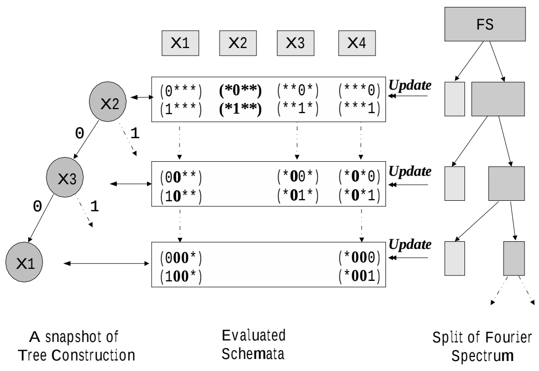

Figure 2: Illustration of the Tree Construction from Fourier Spectrum (TCFS) algorithm

In general, it is not immediately obvious to compare running times of conventional tree induction algorithms and TCFS. In essence, TCFS uses the same criteria to construct a tree as that of the C4.5, which requires a number of information gain test 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 between the two models depends on the sizes of the original data and that of Fourier spectrum.

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.

4 Empirical Study

This section reports the experimental performance of the proposed decision tree aggregation scheme using a semi-synthetic data stream with 100 discrete at-

Block Size	Bagging		Arc-fx	
	#FCs	#Nodes	#FCs	#Nodes
100	103	3008	151	3300
200	148	4288	322	5216
300	256	4904	688	6092

Table 1: The number of FCs that contain 99% and 99.5% of the total energy for Bagging and Arc-fx, respectively.

tributes. The objective is to continuously evolve a decision tree-based predictive model for a particular Boolean attribute using the ensemble approach. 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 this experiment, we assigned 4 discrete values that denote levels of change. 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, which is changed many times during a single experiment. We also added white Gaussian noise to the generator. A test data set of 10,000 instances is generated from the data source prior to the experiment.

We implemented versions of Bagging [3] and Arc-fx (Arcing) [4]. For each case, we built ensembles with different sample block sizes (N). In particular, $N =$ 100, 200 and 300. Each ensemble consists of 100 base models.

Table 1 compares model complexities of original ensembles with their Fourier representations that contain 99% and 99.5% of the total energy for Bagging and Arc-fx respectively. In the case of Arc-fx, we needed more energy to represent the corresponding ensemble. It can be understood that Arc-fx is more diverse (or, possibly more specific) than Bagging in terms of decision paths. However, in either case, the result faithfully illustrates the compactness of the Fourier representation of decision trees. The subsequent experiments are based on these Fourier spectra.

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

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 Size	Confidence		
		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.

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 tree that is constructed from the Fourier spectrum of an ensemble. 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.

5 Conclusion and Future Directions

A novel approach to construct a single decision tree from an ensemble classifier is presented in this paper. We showed that information gain is directly computable from the Fourier spectrum of an ensemble classifier. Subsequently, a fast algorithm that constructs a tree 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. Preliminary experimental results strongly confirm that an informative tree of a reasonable quality can be constructed from a given Fourier spectrum, which is amenable to the discovery of significant patterns.

Ensemble models offer new challenges to knowledge visualization: How can one display the dominant patterns in an ensemble? Traditional decision tree visualization techniques [1] usually focus on visualizing the known structure, thus, not applicable to an ensemble. TCFS algorithm presented here opens a new way to visualize an ensemble. At each level of a tree, TCFS can measure expected information gains quickly so that the user can decide which direction to explore. An individual may easily verify the significance of a rule in which he/she is interested.

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. However, examination of this and similar possibilities will be left for future research.

Acknowledgments

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

References

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

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

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

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

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

[7] Y. Guo and J. Sutiwaraphun. Distributed learning with knowledge probing: A new framework for distributed data mining. In *Advances in Distributed and Parallel Knowledge Discovery, Eds: Hillol Kargupa and Phillip Chan*, pages 115–132. MIT Press, 2000.

[8] 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), January 2002.

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

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

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

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

[13] 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.