System Complexity Reduction via Feature Selection

by

Houtao Deng

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved April 2011 by the
Graduate Supervisory Committee:

George C. Runger, Chair
Sharon L. Lohr
Rong Pan
Muhong Zhang

ARIZONA STATE UNIVERSITY

May 2011

ABSTRACT

This dissertation transforms a set of system complexity reduction problems to feature selection problems. Three systems are considered: classification based on association rules, network structure learning, and time series classification. Furthermore, two variable importance measures are proposed to reduce the feature selection bias in tree models.

Associative classifiers can achieve high accuracy, but the combination of many rules is difficult to interpret. Rule condition subset selection (RCSS) methods for associative classification are considered. RCSS aims to prune the rule conditions into a subset via feature selection. The subset then can be summarized into rule-based classifiers. Experiments show that classifiers after RCSS can substantially improve the classification interpretability without loss of accuracy.

An ensemble feature selection method is proposed to learn Markov blankets for either discrete or continuous networks (without linear, Gaussian assumptions). The method is compared to a Bayesian local structure learning algorithm and to alternative feature selection methods in the causal structure learning problem.

Feature selection is also used to enhance the interpretability of time series classification. Existing time series classification algorithms (such as nearest-neighbor with dynamic time warping measures) are accurate but difficult to interpret. This research leverages the time-ordering of the data to extract features, and generates an effective and efficient classifier referred to as a time series forest (TSF). The computational complexity of TSF is only linear in the length of time series, and interpretable features can be extracted. These features can be further reduced, and summarized for even better

interpretability.

Lastly, two variable importance measures are proposed to reduce the feature selection bias in tree-based ensemble models. It is well known that bias can occur when predictor attributes have different numbers of values. Two methods are proposed to solve the bias problem. One uses an out-of-bag sampling method called OOBForest, and the other, based on the new concept of a partial permutation test, is called a pForest. Experimental results show the existing methods are not always reliable for multi-valued predictors, while the proposed methods have advantages.

To my family

ACKNOWLEDGMENTS

I want to express my deep and sincere gratitude to my advisor, Dr. George Runger, for his generous support and insightful guidance throughout my study at ASU. He offered me great research opportunities, resources, and trust which allowed me to fully explore the research area.

Many thanks to three other members of my dissertation committee: Dr. Sharon Lohr, Dr. Rong Pan, and Dr. Muhong Zhang. They have provided valuable comments and suggestions for both my dissertation and my future career.

I would like to thank all the people who have helped me at ASU. Special thanks to my graduate student colleagues for our collaboration and friendship. I also would like to extend my gratitude to my industry collaborators: Dr. Eugene Tuv, and Dr. Ben Nelson.

Last but not the least, I wish to thank my family, especially my grandparents, parents, and wife for their tremendous love and support. Without their encouragement and support it would have been impossible for me to finish this work.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

Table Page

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Efficient and effective feature selection algorithms have been developed for high-dimensional data sets. This dissertation transforms a set of system complexity reduction problems to feature selection problems, which makes the original problems much easier to solve. Three systems are considered: classification based on association rules, network structure learning, and time series classification. Furthermore, two variable importance measures are proposed to reduce the feature selection bias in tree models.

Both accuracy and interpretability are important for a classifier. Associative classifiers have been shown to be more accurate than decision trees, and each individual rule is interpretable [1–4]. However, the large collection of rules must be combined for a classifier, and this makes it difficult to interpret the classifier. While many algorithms have been proposed for rule pruning, few of these algorithms aimed to select a minimum number of rules without loss of accuracy. Rule condition subset selection (RCSS) considered here aims to select a maximum-relevancy-minimum-redundancy subset of rule conditions, and then summarize the subset into an easily interpretable classifier. While most existing approaches for rule pruning tend to be ad hoc, RCSS is more principled by using well-established feature selection methods.

A Bayesian network is a directed acyclic graph that allows efficient and effective representation of the joint probability distribution over a set of random variables which are represented as nodes in the graph. Markov Blankets (MB) discovery algorithms are

important for learning a Bayesian network structure. Under certain conditions (faithfulness to a Bayesian Network), the MB of a network node is identical to the parents, children, and co-parents of the node [5]. Therefore, the MB of a node is essentially the network local structure of that node. Feature selection methods maximizing relevancy and minimizing redundancy could be used as Markov blanket discovery algorithms, and thus for learning causal structure in networks. [6] and [7] showed the advantages of feature selection methods over a Bayesian network learning algorithm for learning causal structure.

Linear relationship between variables and Gaussian distribution were commonly assumed in learning networks where the variables are continuous. However, the assumptions do not hold in some well-known contexts (e.g, fMRI [8]). A tree ensemble feature selection method is proposed to learn the Markov blankets of networks for either discrete or continuous networks (without linear, Gaussian assumptions) [9]. To learn the Markov blanket of a node in a network, the node is treated as the response variable, other nodes are treated as the predictor variables, then the tree ensemble method can be applied. Experiments conducted here show the tree ensemble method is superior to traditional methods.

Feature selection is also used to enhance the interpretability of time series classification. Time series classification [10–12] has been an active research topic in recent years. Time series classification methods can be divided into instance-based and feature-based methods. Instance-based classifiers predict a test instance based on its similarity to the training instances. One of the most commonly used instance-based classifiers, one-

nearest-neighbor classifiers with a dynamic time warping metric (NNDTW) have been widely, and successfully used. Dynamic time warping (DTW) [13] may be considered simply as a tool to measure the dissimilarity between two time series, after aligning them with an optimal match under certain conditions [14]. DTW is robust to the distortion of time axis, and is considered a strong solution for time series problems [15]. However, it is hard to interpret instance-based classifiers such as NNDTW. Feature-based classifiers generally are more interpretable than instance-based classifiers, as they extract interpretable features, and then input the features to conventional classifiers. However, as shown in Chapter 5, the feature space can be very large, which makes it challenging to select useful features and interpret. This research presents an efficient and effective time series classification framework. Under the framework, interpretable features can be extracted efficiently, and feature selection algorithms can be used to reduce the number of features, which can result in interpretable classifiers.

Variable importance measures for supervised learning are closely related to variable/feature/attribute selection and they are important for improving both learning accuracy and interpretability. Tree ensembles such as random forest are widely used in measuring variable importance [16, 17]. However, the bias problem for multi-valued attributes has been recognized for information-based measures commonly used in tree models. For example, the Gini gain measure is biased in favor of those variables with a larger number of levels [18]. Two methods are proposed to solve the bias problem [19]. One uses an out-of-bag sampling method called OOBForest and one, based on the new concept of a partial permutation test, is called pForest. The experiments show the ex-

isting methods are not always reliable for multi-valued predictors, while the proposed methods have advantages.

In this dissertation, Chapter 1 is an introduction, and Chapter 2 summarizes the background. Chapter 3 describes a framework for integrating associative classification rule condition ranking, pruning and summarizing effectively and efficiently. Chapter 4 describes learning Markov blankets in networks via feature selection. Chapter 5 presents an efficient and interpretable time series classification framework. Chapter 6 proposes two importance measures for reducing feature selection bias in tree models. The conclusions are given in Chapter 7.

CHAPTER 2

BACKGROUND

This chapter summarizes the background of the four topics in this dissertation. The background for each topic is also introduced in each of the following four chapters.

2.1. Associative classification rule condition subset selection

The Apriori algorithm [20] provides an efficient and effective way for mining simple association rules from transaction data. Since then, association rule mining has gained much recognition in the data mining area [21–32]. A rule example such as ([33]):

$$\{peanut\ butter, jelly\} \rightarrow \{bread\} \tag{2.1}$$

indicates that customers that buy $peanut\ butter$ and $jelly$ also buy $bread$. In the rule example, $peanut\ butter, jelly, bread$ are called items. [1] proposed an associative classifier integrating association rule mining and classification (CBA). CBA treats a pair of $< attribute, value >$ as an item, and then association rule algorithms can be used. An associative classification rule such as ([1]): $\{(A = 1, B = 1) \rightarrow y = 1\}$ denotes that the target class is predicted to be 1 if attributes $A = 1$ and $B = 1$. Here $(A = 1, B = 1)$ is also called the rule condition of the classification rule. Rule pruning and summarizing have been active research topics. [34,35] have given reviews in the areas of rule pruning, and rule summarizing for associative classification.

Pruning irrelevant and redundant rules leads to better interpretability and efficiency for classification. Pessimistic error estimation was used for rule pruning in

CBA [1]. Classification based on multiple association rules (CMAR) [3] used Chi-square testing to discard irrelevant rules; [3] pruned specific rules with less confidence values than general rules; [36] removed conflicting rules, where two rules have the same condition, but have different rule outcomes (class), though [37] shows experts could profit from the conflicting rules. [38] proposed a visual and interactive user application for rule pruning.

These pruning methods prune a rule set to some degree, but do not focus on selecting a minimum subset of rules without loss of accuracy. Recently, [39] transformed a set of rules into an indicator data set, then Lasso regression [40] was used to select a small subset of rules. However, a linear model like Lasso does not consider nonlinear relationship between the rules and the class. [41] considered frequent patterns (patterns that occur frequently in data) as features, and proposed a method to select a subset of frequent patterns for classification. The data transformation method used here is the same as [41]. However, [41] focused on only the accuracy of using frequent patterns, while rule condition subset selection (RCSS) considered here focuses on both accuracy and interpretability, e.g. the size of the rule condition subset. Furthermore, the feature selection method proposed in [41] requires a user-defined value to decide the subset size, and is ad hoc, while RCSS applies well-established feature selection algorithms for pruning, and some of them do not require any user-defined value.

In addition to pruning, classification rules need to be summarized for classification. [42] computed statistical significance measures for ordering the rules, and used confidence intervals for the support and confidence of rules to predict future data. The

CBA algorithm [1] generated associative classification rules and built the classifier by gradually adding informative rules to a predictive rule subset. The criteria for choosing rules included support, confidence and the number of data samples covered, etc. CBA only used one rule at a time to predict a new case, while CMAR [3] proposed to use multiple rules at each time for a more robust prediction. Also, predictive association rules (CPAR) [4] extended FOIL (First Order Inductive Learner) [43] to integrate association rules into a classifier. Furthermore, [44] built a classifier according to a rule's information such as confidence and support, and pruned the classifier using pessimistic error estimation similar to decision tree pruning. The method effectively reduced the size of the classifiers without loss of accuracy. HARMONY [45] used an instance-centric rule-generation approach in the sense that it can assure for each training instance, one of the highest-confidence rules covering this instance is included in the rule set. [46] presented a gain based association rule classification (GARC), which could produce a smaller set of rules than such associative classifiers as CBA. [47] proposed a multi-class classification algorithm based on association rules (MCAR), and [48] presented a class-based associative classification approach (CACA). A lazy associative classifier was proposed to generate more rules that are useful for classifying a testing instance [49]. Also, associative classifiers focusing on multi-label problems were studied by [37, 50, 51]. [39] used Lasso regression to form a linear combination of rules for prediction. However, the linear combination of rules are hard to interpret. More recently, [52] summarized associative classification rules using decision trees. Though decision tree is used to summarize rule condition into classifiers in the experiments conducted here, the sum-

marizing framework considered here is more principled and flexible than [52]. Under this summarizing framework, the rule conditions are transformed to an indicator data set, and then any classifiers such as decision trees and naive Bayes classifiers can be chosen for summarization.

2.2. Local structure learning in networks

Bayesian networks are directed acyclic graphs that allow efficient and effective representation of the joint probability distribution over a set of random variables [53]. Linear and Gaussian assumptions were common used in learning networks where the variables are continuous. However, the assumptions do not hold in some well-known contexts (e.g, fMRI [8]). Markov blanket learning can initialize a causal structure learning algorithm, and can be helpful for learning the structure of Bayesian networks. The Markov blanket is defined as follows [54]:

Let F be a full set of variables. Given a target variable T, let $MB(T) \subset F$ and $T \notin MB(T)$, $MB(T)$ is said to be a Markov Blanket (MB) for T if $T \perp (F - MB)|MB$. That is, T is conditionally independent of other features given MB.

Therefore, $MB(T)$ contains all information for predicting T. Under certain conditions (faithfulness to a Bayesian Network), MB(T) is identical to T's parents, its children, and its children's other parents (co-parents) [5]. In the Bayesian network example shown in Figure 1, X_1, X_2 are parents of T, X_3, X_4 are children of T, and X_5, X_6 are co-parents of T. Therefore X_1 to X_6 are the Markov blanket of T. A network struc-

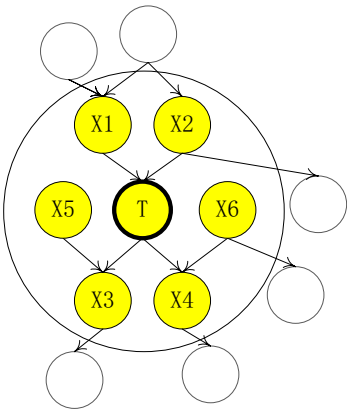


Figure 1. $X_1 \sim X_6$ are the Markov blanket of T in the Bayesian network.

ture learning method [55] first identified the MB of each node, and then all the MBs were used to construct the Bayesian network of the domain. Algorithms (e.g., [5]) have been proposed for identifying MBs in a Bayesian Network. However, most Bayesian network learning algorithms are designed to learn either networks with discrete variables or networks with continuous variables under the Gaussian-distribution, linear-relation assumptions.

MB learning is also closely related to feature selection and [54] stated that MB is an optimal solution for a feature selection method. The MB definition is similar to the maximal relevancy and minimal redundancy principle used in [56], and [57]. There are common characteristics between current MB learning and feature selection algorithms. For example, the feature selection method [57], selected relevant features in a forward phase and removed redundant features in a backward phase (similar to the two phases described in a Markov Blanket learning algorithm [5]).

Therefore, those feature selection methods maximizing relevancy and minimiz-

ing redundancy could be used for learning MBs and thus for learning causal structure in networks. [6] and [7] showed the advantages of feature selection methods over a Bayesian network learning algorithm for learning causal structure. [6] used the features selected from C5.0 rules to identify Markov blankets of a discrete Bayesian network. However, the C5C algorithm is based on only one decision tree and the greedy nature of a single tree could lead to local optimum. [7] applied feature selection utilizing support vector machine based on recursive feature elimination (SVM-RFE) to discover causal structure of continuous networks. This research proposes to use a tree ensemble feature selection algorithm: artificial contrasts with ensembles (ACE) [58], to learn local causal structure in both discrete and continuous Bayesian networks without any distribution assumptions. Tree models can naturally handle mixed categorical and continuous variables, missing values, are robust to outliers in input space, and are insensitive to monotone transformations of inputs [59]. Therefore, ACE can be easily applied to learn network structures in practical applications.

2.3. Time series classification and feature extraction

Time series classification [10–12, 60–62] has been an active research topic in recent years. Time series classification methods can be divided into instance-based and feature-based methods. Instance-based classifiers (e.g. [60, 63]) predict a test instance based on its similarity to the training instances. Among instance-based classifiers, one-nearest neighbor (1NN) classifiers with a Euclidean distance metric (NNEuclidean) or

a dynamic time warping metric (NNDTW) have been widely, and successfully used [63–67]. Usually NNDTW performs better than NNEuclidean since DTW [13] is robust to the distortion of time axis, and is considered as a strong solution for time series problems [15]. A disadvantage of NNDTW is lack of interpretability. To this end, [68] proposed a decision tree approach, which split examples based on dynamic warping distance between a pair of time sequences. Such a decision tree improves interpretability over NNDTW to some degree, and speeds up the time for testing, but the interpretability is still limited.

Feature-based classifiers extract interpretable features, and then input the features to conventional classifiers, which generally are more interpretable than instance-based classifiers. [69] incorporated domain knowledge into an automated search for extracting features, and then applied conventional classifiers such as a support vector machine (SVM) [70]. [71] extracted statistical features such as mean and deviation from time series, and then a multi-layer perceptron neural network was used for classification. [72] extracted features from intervals of time series, and SVM was trained on the features.

[73] extracted features from intervals of time series, and built boosting binary stumps, while [74] used the features from boosting binary stumps to build a single decision tree, which leads to a comprehensible classifier. However, only binary stumps were boosted, and the effect of using more complex base learners, such as decision trees, should be studied [73] (but larger tree models impact the computational complexity of their method). This work shows that this issue can be handled with a simple random

strategy (linear in the length of the time series). More importantly, [73, 74] used only class-based information to evaluate the features. However, the number of extracted features is generally large, sometimes even larger than the number of training examples, and a number of features can provide the same class information (e.g., entropy gain). Consequently, class entropy alone can result in features useful for generalization being ignored.

Advances in feature selection [58], allowed [75] to consider a massive number of features. Features sets were derived from statistical moments, wavelets, Chebyshev coefficients, PCA coefficients, and the original values of the signal. Furthermore, time-warped invariant versions of these features were generated as described by [76]. This massive set of features were used in a dynamic gradient-boosted tree (GBT) ensemble [77] for classification. The dynamic enhancement allows the ensemble to provide higher weight to features with more potential and reduce the computational burden. Performance was excellent in the time series classification challenge [75]. Still, it is computationally complex and features such as PCA scores and Chebyshev coefficients are difficult to interpret and the GBT classifier obfuscates results further. The objective of this work is to produce a fast, accurate classifier that yields a simple set of features that can contribute to the domain knowledge. For example, in manufacturing applications, specific properties of the signals that discriminate conforming from un-conforming product is invaluable to diagnose, correct, and improve processes.

Ensemble methods have been shown effective for time series classification [75, 78, 79]. However, lack of interpretability, and high computational complexity are

disadvantages of the current ensemble methods. Here an effective, yet efficient, ensemble method, referred to as a time series forest (TSF) is proposed for time series classification and feature extraction.

2.4. Bias of importance measures for multi-valued attributes

Variable importance measures for supervised learning are closely related to variable/feature/attribute selection and they are important for improving both learning accuracy and interpretability. However, the bias problem for multi-valued variables has been recognized for well-known variable importance measures such as information-based measures [18]. The number of distinct values of a variables is called cardinality. [80] noted that variable selection with Gini gain measure is biased in favor of those variables with higher cardinality. [18] showed that there are biases in information-based measures adopted by decision tree inductions. [81] showed that variable selection biases not only exist in information-based measures such as the Gini index, but also in others such as the distance measure in Relief [82], etc.

For solving the multi-valued problem, [83] introduced a normalization into the variable selection measure known as the gain ratio. However, attributes with very low information values then appeared to receive an unfair advantage [18, 81]. Also [18] experimented with discrete, uniformly distributed variables with different number of levels. They concluded that Chi-square could be used for the multi-valued problem. [81] proposed a minimum description length principle to alleviate the feature selection bias,

but also mentioned that there are still slight decreases in the importance measure with the increasing cardinality.

Recently, a conditional inference framework [84] was proposed to solve the overfitting and attribute selection bias problems. One method under the framework, cForest [85] demonstrated promising results in their experiments. In addition, [16] introduced a permutation importance measure (PIMP) and demonstrated its advantage over cForest and the original random forest for feature selection. PIMP permutes the response variable and a p-value can be used to measure the variable importance. However, PIMP fits the importance score with a prior probability distributions. Though specifying a prior distribution is not necessary, [16] used prior probability distribution in their experiments (e.g., a gamma distribution for the simulated data, and a normal or lognormal distribution in other cases) and this requires such a prior to be specified in practice. One of the algorithms proposed here: pForest also uses permutation importance, however, there are significant differences between PIMP and pForest. pForest permutes the predictor variables, and more importantly, makes use of a partial permutation strategy for better efficiency. Furthermore, pForest does not need to specify a prior probability distribution.

CHAPTER 3

ASSOCIATIVE CLASSIFICATION RULE CONDITION SUBSET SELECTION

3.1. Introduction

Both accuracy and interpretability are important for a classifier. Associative classifiers have been shown to be more accurate than decision trees, and each individual rule is interpretable [1–4]. However, the large collection of rules must be combined for a classifier, and this makes it difficult to interpret the classifier. While many algorithms have been proposed for pruning and summarizing, few of these algorithms aimed to select a minimum number of rules without loss of accuracy. Rule condition subset selection (RCSS) considered here aims to select a maximum-relevancy-minimum-redundancy subset of rule conditions, and then summarize the subset into an easily interpretable classifier. While existing approaches for rule pruning tend to be ad hoc, RCSS is more principled by using well-established feature selection methods.

The remainder of this chapter is organized as follow. Section 3.2 introduces previous work related to rule pruning and summarizing. Section 3.3 describes how to form a new data set, and how supervised learning methods are applied to rank, prune and summarize conditions. Section 3.4 demonstrates the effectiveness and efficiency of RCSS by testing on data sets from UCI repository [86]. Conclusions are presented in Section 3.5.

3.2. Related work

The Apriori algorithm [20] provides an efficient and effective way for mining simple association rules from transaction data. Association rule mining has gained much recognition in the data mining area [21–32]. A rule example such as ([33]):

$$\{peanut\ butter, jelly\} \rightarrow \{bread\} \tag{3.1}$$

indicates that customers that buy $peanut\ butter$ and $jelly$ also buy $bread$. In the rule example, $peanut\ butter, jelly, bread$ are called items. [1] proposed an associative classifier integrating association rule mining and classification (CBA). CBA treats a pair of $< attribute, value >$ as an item, and then association rule algorithms can be used. An associative classification rule such as ([1]): $\{(A = 1, B = 1) \rightarrow y = 1\}$ denotes that the target class is predicted to be 1 if attributes $A = 1$ and $B = 1$. Here $(A = 1, B = 1)$ is also called the rule condition of the classification rule, which is generally a conjunction of attribute-value pairs. Since then, rule pruning and summarizing have been active research topics. [34, 35] have given reviews in the areas of rule pruning, and rule summarizing for associative classification.

Pruning irrelevant and redundant rules leads to better interpretability and efficiency for classification. Pessimistic error estimation was used for rule pruning in CBA [1]. Classification based on multiple association rules (CMAR) [3] used Chi-square testing to discard irrelevant rules; [3] pruned specific rules with less confidence values than general rules; [36] removed conflicting rules, where two rules have the same condition, but have different rule outcomes (class), though [37] shows experts could

profit from the conflicting rules. [38] proposed a visual and interactive user application for rule pruning.

These pruning methods prune a rule set to some degree, but do not focus on selecting a minimum subset of rules without loss of accuracy. Recently, [39] transformed a set of rules into an indicator data set, then Lasso regression [40] was used to select a small subset of rules. However, a linear model like Lasso does not consider nonlinear relationship between the rules and the class. [41] considered frequent patterns (patterns that occur frequently in data) as features, and proposed a method to select a subset of frequent patterns for classification. The data transformation method used here is the same as [41]. However, [41] focused on only the accuracy of using frequent patterns, while rule condition subset selection (RCSS) considered here focuses on both accuracy and interpretability, e.g. the size of the rule condition subset. Furthermore, the feature selection method proposed in [41] requires a user-defined value to decide the subset size, and is ad hoc, while RCSS applies well-established feature selection algorithms for pruning, and some of them do not require any user-defined value.

In addition to pruning, classification rules need to be summarized for classification. [42] computed statistical significance measures for ordering the rules, and used confidence intervals for the support and confidence of rules to predict future data. The CBA algorithm [1] generated associative classification rules and built the classifier by gradually adding informative rules to a predictive rule subset. The criteria for choosing rules included support, confidence and the number of data samples covered, etc. CBA only used one rule at a time to predict a new case, while CMAR [3] proposed to use mul-

tiple rules at each time for a more robust prediction. Also, predictive association rules (CPAR) [4] extended FOIL (First Order Inductive Learner) [43] to integrate association rules into a classifier. Furthermore, [44] built a classifier according to a rule's information such as confidence and support, and pruned the classifier using pessimistic error estimation similar to decision tree pruning. The method effectively reduced the size of the classifiers without loss of accuracy. HARMONY [45] used an instance-centric rule-generation approach in the sense that it can assure for each training instance, one of the highest-confidence rules covering this instance is included in the rule set. [46] presented a gain based association rule classification (GARC), which could produce a smaller set of rules than such associative classifiers as CBA. [47] proposed a multi-class classification algorithm based on association rules (MCAR), and [48] presented a class-based associative classification approach (CACA). A lazy associative classifier was proposed to generate more rules that are useful for classifying a testing instance [49]. Also, associative classifiers focusing on multi-label problems were studied by [37, 50, 51]. [39] used Lasso regression to form a linear combination of rules for prediction. However, the linear combination of rules are hard to interpret. More recently, [52] summarized associative classification rules using decision trees. Though decision tree is used to summarize rule condition into classifiers in the experiments conducted here, the summarizing framework considered here is more principled and flexible than [52]. Under this summarizing framework, the rule conditions are transformed to an indicator data set, and then any classifiers such as decision trees and naive Bayes classifiers can be chosen for summarization.

3.3. Associative classification rule analysis based on rule conditions

A training data set is denoted as $D = \{(x_i, y_i)|i = 1...n\}$, where $y_i \in Y$, where Y is a set of class labels: $\{1, 2, ...|Y|\}$, and $|Y|$ is the number of class values. A classification rule r_j describing D can be expressed as $\{c_j \rightarrow y = k\}$, where $k \in Y$ and c_j is the condition of r_j.

Given a set of classification rules describing D: $\{r_1, r_2, ..., r_m\}$, instead of directly pruning and summarizing the rules, this work considers pruning and summarizing the rule conditions: $\{c_1, c_2, ..., c_m\}$ after transforming the rule conditions to a new data set. The same technique was used for transforming a set of rules [39] or a set of frequent patterns [41] to an indicator data set.

3.3.1. Transform conditions to an indicator data set

Let I_{ij} be an indicator variable that denotes whether a condition $c_j \in \{c_1, c_2, ..., c_m\}$ is satisfied for the predictors in a training instance (x_i, y_i): x_i. That is,

$$I_{ij} = \begin{cases} 1 & c_j \text{ is satisfied for } x_i \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

A new data set I is formed based on the indicator variables and the class labels: $\{[I_{i1}, ..., I_{im}, y_i], i = 1, ...n\}$. Denote the attributes (excluding the class) as $\{I_1, I_2, ..., I_m\}$. The conditions: $\{c_1, c_2, ..., c_m\}$ are represented by the attributes:

$\{I_1, I_2, ..., I_m\}$ in I and directly linked to the class variable y. Thus the dependency between the conditions and the class can be measured.

Take the XOR logical operation as an illustrative example. There are four logical operations for XOR. Replicate each row 40 times to form a 160-row data set as shown in Table 1. Then extract associative rules from the data set (with minimum support greater than 0). As shown in Table 2, a total of 14 rules are generated, with their support and confidence shown in the table. The new data set I is formed from the data in Table 1 and the conditions in Table 2 and are shown in Table 3.

TABLE 1
40 cases are simulated for each XOR operation.

data row ID	X_1	X_2	y
1-40	0	0	0
41-80	0	1	1
81-120	1	0	1
121-160	1	1	0

TABLE 2
14 associative classification rules generated from the XOR data.

Rule ID	Conditions	Class	Support	Confidence
R1	c_1: {}	y=0	0.5	0.5
R2	c_2: X1=0	y=0	0.25	0.5
R3	c_3: X2=0	y=0	0.25	0.5
R4	c_4: X2=1	y=0	0.25	0.5
R5	c_5: X1=1	y=0	0.25	0.5
R6	c_6: X1=0,X2=0	y=0	0.25	1
R7	c_7: X1=1,X2=1	y=0	0.25	1
R8	c_8: {}	y=1	0.5	0.5
R9	c_9: X1=0	y=1	0.25	0.5
R10	c_{10}: X2=0	y=1	0.25	0.5
R11	c_{11}: X2=1	y=1	0.25	0.5
R12	c_{12}: X1=1	y=1	0.25	0.5
R13	c_{13}: X1=0,X2=1	y=1	0.25	1
R14	c_{14}: X1=1,X2=0	y=1	0.25	1

3.3.2. Rank conditions

For a classification problem, the importance of an attribute should be measured by the dependency between the attribute and the class. In classification rule analysis,

TABLE 3
New data formed from the data in Table 1 and the conditions in Table 2

data row ID	I_1	I_2	$\cdots$	I_{14}	y
1-40	1	1	$\cdots$	0	0
41-80	1	1	$\cdots$	0	1
81-120	1	0	$\cdots$	1	1
121-160	1	0	$\cdots$	0	0

statistics such as the support and confidence of a rule are also useful criteria for evaluating the rule importance. Thus, both support and confidence are usually considered in associative classifiers. For example, CBA [1] sorts the rules by confidence and support, and then considers the dependency of the rules and the class. In this framework, rule condition ranking is achieved at one time by applying an importance measure on I. Measuring the importance of an attribute I_j in I is essentially measuring the importance of the condition c_j to the class. In fact, because I includes information (such as support and confidence statistics) on the rules the importance measure could be a function of these statistics.

Any methods for measuring attribute importance in supervised learning area can be used to measure the importance of conditions here. Symmetric Uncertainty (SU) [87] and random forest (RF) [88] are considered here. First consider SU. The entropy $H(y)$ of the class variable y is

$$H(y) = -\sum_{k=1}^{|Y|} P(y=k) log_2(P(y=k))$$

The information gain IG of y after observing I_j is

$$IG(y|I_j) = H(y) - H(y|I_j)$$

Then SU is defined as

$$SU(I_j, y) = 2\left[\frac{IG(y|I_j)}{H(I_j) + H(y)}\right] \tag{3.3}$$

$SU(I_j, y)$ is closely related to rule statistics, because $SU(I_j, y)$ can be easily written as a function of the support and confidence of all the rules with c_j as the condition: $\{c_j \to y = k, k = 1, 2, ..., |Y|\}$.

Although SU measures the dependency ($dependency(c_j, y)$) between a condition and the class, it does not consider the effect of other conditions. To measure the dependency ($dependency(c_j, y|c_{i\neq j})$) considering other conditions, an importance measure from tree models can be applied. Interactions between attributes are included in tree models. Furthermore, to overcome the greedy nature of a single decision tree, evaluation from an ensemble is more stable. Thus the importance measure from RF is used. RF uses the Gini index information criteria [88]: $\sum_{k_1 \neq k_2} P(y = k_1)P(y = k_2)$, where k_1 and k_2 run through all class labels.

Still consider the example XOR. In Table 2, if only a measure $dependency(c_j, y)$ is considered, then only $\{c_6, c_7, c_{13}, c_{14}\}$ are useful for predicting the class. If a measure such as $dependency(c_j, y|c_{i\neq j})$ is used, besides the previous four conditions, other conditions such as c_2 and c_3 should be important too, because a combination of c_2 and c_3 can be combined to accurately predict the class.

For predicting the class, the interactions between conditions should be considered, thus $dependency(c_j, y|c_{i\neq j})$ is an important measure.

The importance scores of the conditions using SU and RF (with 1000 trees) are shown in the Table 4. As expected, SU ranks $\{c_6, c_7, c_{13}, c_{14}\}$ as the most important and scores all other conditions with zero importance. RF not only ranks the four conditions as the most important, it also scores some other conditions such as c_2 and c_3 as important.

TABLE 4
Importance score from SU and RF for conditions from Table 2. SU ranks $\{c_6, c_7, c_{13}, c_{14}\}$ as most important and scores all other conditions with zero importance. RF not only ranks the four conditions as most important, it also scores some other conditions such as c_2 and c_3 as important.

ID	SU	RF	ID	SU	RF
c_1	0.00	0.00	c_8	0.00	0.00
c_2	0.00	3.58	c_9	0.00	3.60
c_3	0.00	4.40	c_{10}	0.00	3.39
c_4	0.00	4.44	c_{11}	0.00	3.77
c_5	0.00	4.10	c_{12}	0.00	3.94
c_6	0.34	10.83	c_{13}	0.34	10.29
c_7	0.34	10.56	c_{14}	0.34	9.33

3.3.3. Prune conditions

For better classification efficiency and interpretability, the conditions should be pruned into a smaller-size subset. Though the size is smaller, the subset should still reserve all the information of the original conditions for predicting the class. This is

similar to the goal of feature selection (FS) algorithms (for example, correlation-based feature selection (CFS) [89], fast correlation-Based filter (FCBF) [90], artificial contrasts with ensembles (ACE) [58]) that seek to select a subset of minimum-redundancy-maximum-relevancy attributes for supervised learning, which can be described in terms of a Markov blanket [54].

Definition 1. *Let F be a full set of features. Given a class variable y, let $M \subset F$ and $y \notin M$, M is said to be a Markov blanket (MB) of F for y if $y \perp (F - M)|M$. That is, M is the only subset of features needed to predict y.*

A MB of a set of variables might be considered as a feature selection solution [54]. This work applies the same concept to the pruning of conditions. The goal of condition pruning is to find a MB of the condition set.

Calculating an accurate MB is computationally infeasible since there are $2^m - 1$ subsets for evaluation. Thus, heuristic methods are used to find an approximate MB. By applying feature selection to find an approximate MB of $\{I_1, ..., I_m\}$, a compact subset of conditions that preserve most of the information useful for prediction can be found. An advantage of the feature selection methods is that they are developed for high-dimensional data sets. Therefore, the condition pruning here becomes as efficient as the selected feature selection method.

In the data set I, there may be some attributes (columns) with the same values. These attributes have the same information for predicting the class and thus only one attribute from them is needed to be kept for classification. Keep the attribute I_i whose

corresponding c_i has the simplest form, that is, c_i has the least number of attribute-value pairs among the conditions. This simple pruning serves as the first step before using feature selection method. Then a feature selection method can be applied (e.g., CFS [89], FCBF [90], ACE [58]). For selecting the relevant features, CFS and FCBF use SU as the relevancy measure, and ACE uses the RF measure along with an iterative algorithm. To eliminate redundant conditions, SU is again used by CFS and FCBF to measure redundancy. The class variable y is replaced with a rule condition I_j in equation 3.3 for this role

$$SU(I_i, I_j) = 2\left[\frac{IG(I_i|I_j)}{H(I_i) + H(I_j)}\right] \tag{3.4}$$

Here $SU(I_i, I_j)$ evaluates the dependency between I_i and I_j and a large value indicates that I_i or I_j is redundant.

CFS, FCBF, ACE are applied on I for the XOR example. Denote the data set obtained after using a FS algorithm on I as I^*. The subsets selected by the three methods are shown in Table 5. CFS, FCBF select the conditions that correspond to the four XOR logical rules. The four conditions can predict the class with 100% accuracy. ACE selects only two conditions, which can be used to predict the class correctly in a tree model as well.

3.3.4. Summarize conditions

For associative classification, one important task is to summarize the rules into a classifier. Here a summary of the conditions is needed for classification. In the rule

TABLE 5
The condition subsets selected by CFS, FCBF, ACE.

Method	Condition subset
CFS	c_6, c_7, c_{13}, c_{14}
FCBF	c_6, c_7, c_{13}, c_{14}
ACE	c_6, c_7

TABLE 6
The classifier formed by applying C4.5 to the condition subset selected by ACE in Table 5.

If c_7 is satisfied, then y = 0;
else,
    If c_6 is satisfied, then y = 0;
    else, then y = 1;

analysis area, designing a rule summary method is non-trivial. Not only do a large number of redundant rules need to be organized for fitting the training data, but also pruning needs be considered to avoid possible overfitting. However, in the framework considered here, the design of such a classifier is rather simple and flexible. By applying a classifier to I^*, the conditions can be organized into a classifier easily. Moreover, because I^* has fewer attributes than I, but has similar information regarding the class, applying a classifier on I^* could have better efficiency and interpretability. [41] used support vector machine (SVM) and C4.5 for summarizing frequent patterns, but only focused on classification accuracy.

The option of a large number of classifiers provides a flexible representation for the condition summary. A classifier with both reasonable accuracy and interpretability should be considered so that the model built from the conditions can be understood. For example, by building a decision tree on I^*, a decision is made at each node. If $I_j = 1$ (if condition c_j is satisfied), then go to left branch; else, go to the right branch. The class is assigned at the leaf nodes. For the XOR example, C4.5 is applied to the data

set I^* formed by ACE, a classifier built with conditions is shown in Table 6. Only two conditions are used in classification, and the classifier correctly captures the true pattern. Furthermore, the classifier shown in Table 6 can be simply interpreted as a decision tree that uses conjunction of attributes in the splitting rules (such as c_6: $X_1 = 0, X_2 = 0$).

3.4. Experiments

To demonstrate the efficiency and effectiveness of the methods considered here, data sets from the UCI Repository are analyzed. The R package [91] and RWeka [92,93] are used for running all the experiments, except ACE [58] is programmed in C. The default parameter setting for the methods in the software is used unless specified otherwise. Associative classification rules are extracted with minimum confidence = 0.6, minimum support = 0.05, which are the minimum values allowed in RWeka to produce valid results for all data sets. Furthermore, this work investigates the sensitivity to the minimum support and confidence levels. For generating the association rules, first discretize the continuous attributes using the entropy method [94]. All the following results are obtained from 10-fold cross validation. The condition-based classifiers are compared to C4.5 [95] (pruned tree with one as the minimum number of instances per leaf node) and a well-known associative classifier (CBA [1]) regarding both classification accuracy and interpretability. The results from CBA are obtained from [1] with minimum confidence = 0.5 and minimum support = 0.01. To evaluate the quality of the subsets selected by the FS algorithms, tree classifiers are built on the condition subsets selected and the

accuracy and interpretability of the classifiers are used for the evaluation.

3.4.1. Condition pruning and classification

CFS [89], FCBF [90] and ACE [58] are used to prune the conditions. Because ACE is tuned for datasets with two classes, only two-class results for ACE are shown. The results are shown Table 7. Columns 1-3 provide the name of the data set, the number of classes, and the number of associative classification rules generated, respectively. Columns 4-6 are the numbers of conditions selected after pruning (approximate MBs) from the three feature selection algorithms. All algorithms select a compact subset of conditions with many fewer conditions than those generated from the association rules. FCBF and ACE (for two-class data) select a subset with even smaller size than CFS.

To summarize the conditions into a classifier, C4.5 [95] is applied to I and I^* for all the data sets and then the interpretability and the accuracy of the classification are evaluated. In tree classifiers, the path from the root node to a leaf node can be considered as a rule. Although a reduced set of rules can sometimes be obtained from a tree model, that refinement is not considered here. Therefore, the number of leaf nodes in tree classifiers and the number of rules in CBA are used to evaluate a classifier's interpretability. In Table 7 the remaining columns show the number of leaf nodes in C4.5 trees from the original data sets (Orig), discretized data sets (Disc), full set of conditions (Full), and conditions after pruning with algorithms CFS, FCBF, ACE, and the number of rules in the CBA classifier. Note the substantial reduction in the number of

leaf nodes from trees generated from conditions pruned by CFS, FCBS and ACE based on either the mean or median over the data sets. Compared to trees built on the original data the trees built from the conditions benefit from attributes combined into conditions, but the FS methods still reduce the complexity compared to trees built from the full set of conditions. Furthermore, the number of leaf nodes after FS are substantially fewer than the number of rules used in CBA. However, the quality of the subsets still needs to be evaluated by the classification accuracy.

For the accuracy of the classification, error rates from C4.5 are used for evaluation. Because the objective here is an interpretable model, a simple classifier is applied to the selected conditions. To compare with more complex classifiers, RF with 500 trees is applied to the original data (RFOrig) and discretized data (RFDisc). Figures 2 and 3 show the box plots of error rates from all the methods for two-class data sets and all data sets, respectively. The error rate details are also shown in Table 8.

First consider the interpretable classifiers. The tree-based classifiers pruned with FS have error rates comparable to those from the original or discretized data with respect to both means and medians. There is no significant difference in the mean accuracy between C4.5Orig and C4.5Disc and any of the FS methods at a 5% significance level. The accuracy from the FS methods is also closely comparable to CBA. Only the difference between FCBF and CBA is significant and FCBF uses a substantially less complex model. Also, for the two-class problems, ACE provides excellent accuracy scores (no significant differences with paired t-tests) and compact models.

Although RFOrig provides slightly better performance than the other methods,

the conclusions change once discretized data are considered. That is, there is no significance difference between the FS method CFSC4.5 and RFDisc nor is there a significance difference between ACEC4.5 and RFDisc for the two-class cases. Only FCBFC4.5 is significantly weaker than RFDisc. Consequently, two of the FS methods provide interpretable models that are similar to RF for the discretized data in these examples. Furthermore, a better discretizer might narrow the small gap to the results from RFOrig.

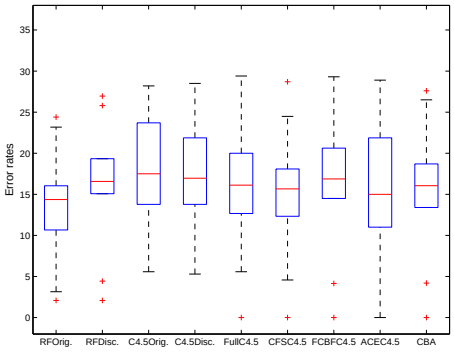


Figure 2. The box plot of error rates from data sets with only two classes

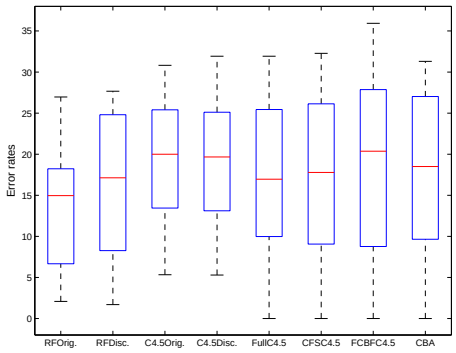


Figure 3. The box plot of error rates from all data sets

In summary, conditions pruned with a FS algorithm can substantially reduce the number of conditions and these conditions can be used in a simple decision tree model that still maintains the accuracy compared to an associative classifier such as CBA (or trees from the full set of conditions). The FS-based CFSC4.5 and ACE4.5 algorithms are competitive with CBA regarding the accuracy, but have many fewer leaf nodes than CBA rules. Two of the FS methods considered here can provide interpretable models that perform as well as an RF applied to the same discretized data. The FS approach is based on well-known, and widely-available FS algorithms that are easy to apply. Also,

the FS methods are not tuned for optimum performance. Potentially these methods could be further improved.

TABLE 7

Two results are shown: 1) The number of conditions after pruning with algorithms CFS, FCBF, ACE 2) The number of leaf nodes in C4.5 trees from the original data sets, discretized data sets, full set of conditions, and conditions after pruning with algorithms CFS, FCBF, ACE and the number of rules in the CBA classifier

			♯ of conditions selected			♯ leaf nodes in trees and ♯ rules in CBA						
Dataset	♯ classes	♯ original rules	CFS	FCBF	ACE	Orig	Disc	Full	CFS	FCBF	ACE	CBA
austra	2	1993	30	8	8	31	10	13	8	5	5	148
auto	7	12626	44	14	-	70	108	25	24	16	-	54
breast	2	894	18	15	16	16	16	13	9	7	9	49
crx	2	5499	44	10	9	24	23	31	11	5	7	142
german	2	4904	50	7	10	115	110	109	21	3	12	172
glass	7	263	13	5	-	33	30	19	15	8	-	27
heart	2	957	31	9	6	24	13	12	8	6	5	52
hepati	2	497	13	6	5	12	5	7	6	4	4	23
horse	2	1221	27	9	6	9	5	10	4	3	4	97
iris	3	64	4	3	-	6	4	4	3	3	-	5
labor	2	742	19	14	20	5	4	5	5	5	5	12
led7	10	193	38	9	-	40	40	28	28	15	-	71
lymph	4	704	15	7	-	18	10	9	7	6	-	36
pima	2	174	17	4	8	19	12	8	7	4	6	45
tic-tac	2	349	11	10	35	115	115	9	9	9	9	8
vehicle	4	6904	32	7	-	85	184	78	33	8	-	125
waveform	3	400	33	20	-	337	692	427	235	150	-	386
wine	3	5382	24	16	-	5	12	4	4	5	-	10
zoo	7	4098	19	15	-	9	8	7	8	9	-	7
mean	-	2519.2	25.4	9.8	-	51.1	73.8	43.0	23.4	14.2	-	77.3
median	-	894.1	24.1	8.9	-	23.6	13.1	12.0	8.4	6.0	-	49.0
mean(2-class)	-	1723.0	26.0	9.2	12.3	36.9	31.3	21.7	8.8	5.1	6.5	74.8
median(2-class)	-	925.6	23.0	8.9	8.6	21.5	12.6	11.0	8.2	4.8	5.4	50.5

TABLE 8

The error rates from C4.5 applied to the original data, discretized data, the full set of conditions, the pruned conditions selected by CFS, FCBF, ACE, and CBA. The last two columns are error rates from random forest applied to the original data (RFOrig) and the discretized data (RFDisc).

Dataset	C4.5Orig	C4.5Disc	FullC4.5	CFSC4.5	FCBFC4.5	ACEC4.5	CBA	RFOrig	RFDisc
austra	16.4	12.9	13.8	14.8	14.5	14.6	13.4	12.9	15.1
auto	20.7	22.0	27.0	30.9	28.4	-	27.2	16.2	18.6
breast	5.6	5.3	5.6	4.6	4.1	3.9	4.2	3.1	4.4
crx	14.9	15.7	17.0	16.4	16.1	15.4	14.1	13.8	15.8
german	28.2	28.5	29.4	28.7	29.3	28.9	26.5	24.4	25.8
glass	30.8	25.6	25.7	27.6	32.2	-	27.4	18.7	25.2
heart	23.7	18.1	15.9	17.8	20.4	19.3	18.5	15.6	18.1
hepati	18.6	21.9	20.0	18.1	20.6	21.9	15.1	16.0	19.3
horse	13.3	15.8	16.3	15.0	15.0	13.9	18.7	15.0	17.1
iris	5.3	6.0	6.7	6.7	6.7	-	7.1	5.3	6.0
labor	20.0	19.7	12.7	12.3	17.7	11.0	17.0	10.7	16.0
led7	25.9	25.9	26.2	26.7	29.7	-	27.8	27.0	27.0
lymph	25.1	23.7	20.9	21.0	25.7	-	19.6	17.0	23.6
pima	25.5	25.1	24.7	24.5	26.2	25.1	27.6	23.2	27.0
tic-tac	13.8	13.8	0.0	0.0	0.0	0.0	0.0	2.1	2.1
vehicle	27.7	31.9	31.9	32.3	35.9	-	31.3	24.4	27.7
waveform	22.7	25.0	23.2	21.9	22.7	-	20.6	14.9	15.6
wine	7.4	9.5	9.1	8.0	6.9	-	8.4	2.3	1.7
zoo	5.9	5.9	4.9	5.0	3.0	-	5.4	3.0	4.0
mean	18.50	18.54	17.41	17.48	18.68	-	17.36	13.97	16.32
mean(2-class)	18.00	17.67	15.53	15.21	16.38	15.39	15.51	13.67	16.08
median	20.00	19.67	16.96	17.78	20.37	-	18.50	14.96	17.13
median(2-class)	17.50	16.96	16.12	15.67	16.88	15.00	16.05	14.37	16.57

3.4.2. Sensitivity to support and confidence

Here investigate the sensitivity of the FS classifiers to minimum support and the minimum confidence changes. First fix the minimum support = 0.05 and analyze different minimum confidence values: $\{0.6, 0.65, 0.7, 0.75, 0.8\}$. then fix the minimum confidence = 0.6 and analyze different minimum support values: $\{0.05, 0.055, 0.06$ $0.065, 0.07, 0.075, 0.08\}$. The average error rates and the number of leaf nodes over all the data sets for these values are shown in Figures 4 to 11. For two-class data sets the average error rates and leaf nodes numbers are stable for the values considered here, except that CFSC4.5 and FCBFC4.5 have obvious error rate increases when the minimum support changes to 0.8 (Figure 6). For all data sets, the average error rates tend to be larger and the leaf nodes numbers tend to be smaller as the minimum support or the minimum confidence increases. This implies that the condition based classifiers are not sensitive to the minimum confidence and minimum support values considered here for two-class problems. However, for the data sets more than two classes, the minimum support and confidence should made small for better classification accuracy.

3.5. Conclusions

Associative classification rule pruning and summarizing have been active research topics. While previous pruning methods focused on pruning rules to some degree, this research proposes to prune rule conditions into a minimum subset without loss information regarding predicting the class, which suggests a new direction for rule

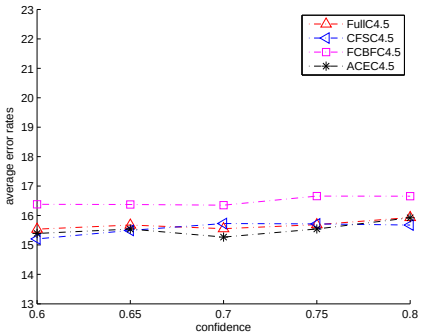


Figure 4. The average error rates of tree classifiers for two-class data sets as minimum confidence changes with fixed minimum support = 0.05

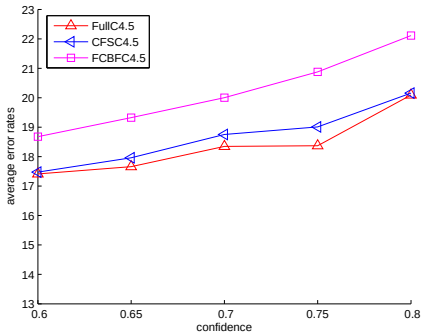


Figure 5. The average error rates of tree classifiers for all data sets as the minimum confidence changes with fixed minimum support = 0.05

pruning. In addition, although previous associative classifiers were derived from simple association rules, the large number of rules used in the final model results in limited interpretability. The rule condition based classifiers discussed here consist of a significantly smaller number of rules without loss of accuracy, comparing to previous well-known associative classifiers in the experiments conducted here.

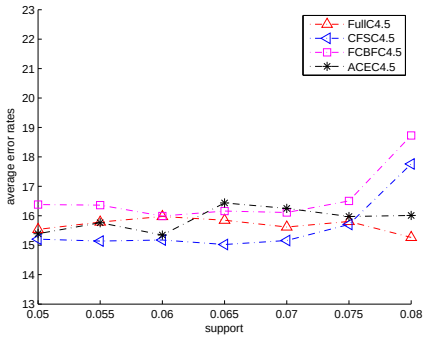


Figure 6. The average error rates of tree classifiers for two-class data sets as the minimum support changes with fixed minimum confidence = 0.6

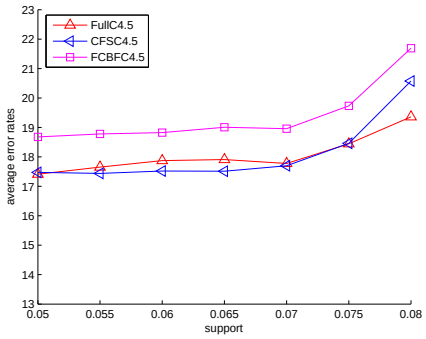


Figure 7. The average error rates of tree classifiers for all data sets as the minimum support changes with fixed minimum confidence = 0.6

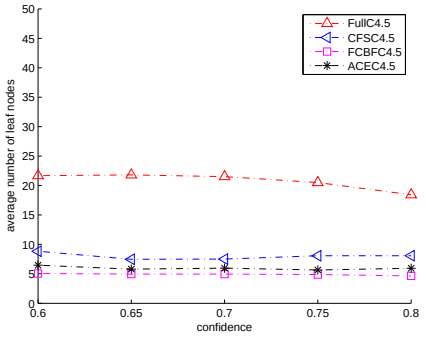


Figure 8. The average number of leaf nodes in tree classifiers for two-class data sets as the minimum confidence changes with fixed minimum support = 0.05

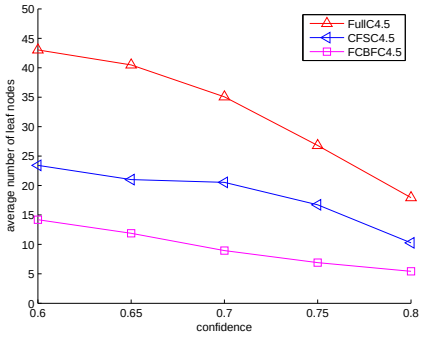


Figure 9. The average number of leaf nodes in tree classifiers for all data sets as the minimum confidence changes with fixed minimum support = 0.05

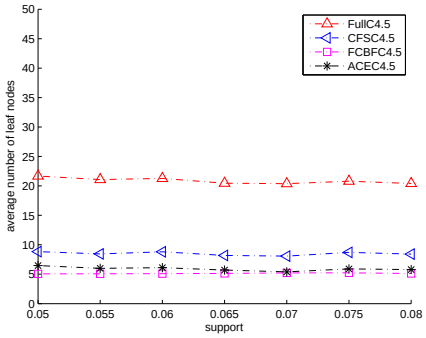


Figure 10. The average number of leaf nodes in tree classifiers for two-class data sets as the minimum support changes with fixed minimum confidence = 0.6

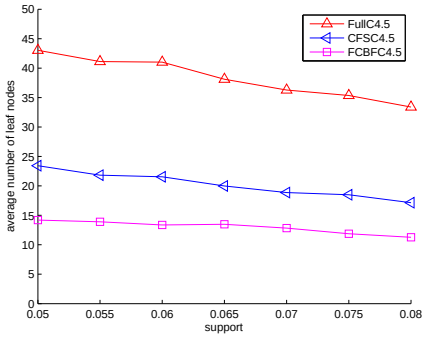


Figure 11. The average number of leaf nodes in tree classifiers for all data sets as the minimum support changes with fixed minimum confidence = 0.6

CHAPTER 4

LEARNING MARKOV BLANKETS FOR CONTINUOUS OR DISCRETE NETWORKS VIA FEATURE SELECTION

4.1. Introduction

Bayesian networks are directed acyclic graphs that allow efficient and effective representation of the joint probability distribution over a set of random variables [53]. Linear and Gaussian assumptions were common used in learning networks where the variables are continuous. However, the assumptions do not hold in some well-known contexts (e.g, fMRI [8]). Markov blanket learning can initialize a causal structure learning algorithm, and can be helpful for learning the structure of Bayesian networks. The Markov blanket is defined as follows [54]:

Let F be a full set of variables. Given a target variable T, let $MB(T) \subset F$ and $T \notin MB(T)$, $MB(T)$ is said to be a Markov Blanket (MB) for T if $T \perp (F - MB)|MB$. That is, T is conditionally independent of other features given MB.

Therefore, $MB(T)$ contains all information for predicting T. Under certain conditions (faithfulness to a Bayesian Network), MB(T) is identical to T's parents, its children, and its children's other parents (co-parents) [5]. A network structure learning method [55] first identified the MB of each node, and then all the MBs were used to construct the Bayesian network of the domain. Algorithms (e.g., [5]) have been proposed for identifying MBs in a Bayesian Network. However, most Bayesian network learning algorithms are designed to learn either networks with discrete variables or networks with continuous variables under the Gaussian-distribution, linear-relation assumptions.

MB learning is also closely related to feature selection and [54] stated that MB is an optimal solution for a feature selection method. The MB definition is similar to the maximal relevancy and minimal redundancy principle used in [56], and [57]. There are common characteristics between current MB learning and feature selection algorithms. For example, the feature selection method [57] selected relevant features in a forward phase and removed redundant features in a backward phase (similar to the two phases described in a Markov Blanket learning algorithm [5]).

Therefore, those feature selection methods maximizing relevancy and minimizing redundancy [57, 58, 89, 90, 96] can be used for learning MBs and thus for learning causal structure in networks. [6] and [7] showed the advantages of feature selection methods over a Bayesian network learning algorithm for learning causal structure. [6] used the features selected from C5.0 rules to identify Markov blankets of a discrete Bayesian network. However, the C5C algorithm is based on only one decision tree and the greedy nature of a single tree could lead to local optimum. [7] applied feature selection utilizing support vector machine based on recursive feature elimination (SVM-RFE) to discover causal structure of continuous networks.

This research proposes to use artificial contrasts with ensembles (ACE) [58], a tree ensemble feature selection algorithm to learn local causal structure in both discrete and continuous Bayesian networks without any distribution assumptions. Tree models can naturally handle mixed categorical and continuous variables, missing values, are robust to outliers in input space, and are insensitive to monotone transformations of inputs [59]. Therefore, ACE can be easily applied to learn network structures in practical

applications.

Section 4.2 describes the feature selection approach. Section 4.3 provides experiments for local structure learning in continuous networks for both linear and nonlinear models and with and without Gaussian assumptions. An example is provided for one discrete Bayesian network. Section 4.4 provides conclusions.

4.2. Feature selection framework

The framework of ACE [58] is outlined in Algorithm 1 (shown for a regression problem) and with notation summarized in Table 9. A similar algorithm applies to classification problems. Several iterations of feature selection are considered to include features important, but possibly weaker than a primary set. In each iteration only the important features are used to predict the target and generate residuals (targets minus model predictions for regression). In subsequent iterations the feature selection is applied to the residuals. However, all variables are input to the feature selection module that builds the ensembles—not only the currently important ones. This is to recover partially masked variables that still contribute predictive power to the model. This can occur after the effect of a masking variable is completely removed, and the partial masking is eliminated. Based on important features, the redundancy elimination module selects a non-redundant feature subset. Brief comments for the functions *SelectFeatures* and *RemoveRedundant* are provided below and further details were provided by [58].

Algorithm 1 Ensemble-Based Feature Selection.

1. Set $\Phi \leftarrow \{\}$; set $F \leftarrow \{X_1, \ldots, X_M\}$; set $I = 0$ ($|I| = M$)
2. Set $[\hat{\Phi}, \Delta I] = SelectFeatures(F, T)$
3. Set $\hat{\Phi} = RemoveRedundant(\hat{\Phi})$
4. If $\hat{\Phi}$ is empty, then quit
5. $\Phi \leftarrow \Phi \cup \hat{\Phi}$
6. $I(\hat{\Phi}) = I(\hat{\Phi}) + \Delta I(\hat{\Phi})$
7. $T = T - g_T(\hat{\Phi}, T)$
8. Go to 2.

TABLE 9
Notation in Algorithm 1

F	set of original variables
T	target variable
M	Number of variables
I	cumulative variable importance vector
Φ	set of important variables
ΔI	current vector of variable importance scores from an ensemble
$\Delta I(\hat{\Phi})$	current variable importance scores for the subset of variables $\hat{\Phi}$
$g_T(F, T)$	function that trains an ensemble based on variables F and target T, and returns a prediction of T

4.2.1. Feature importance measure

Relevant feature selection is based on an ensemble of decision tees. Trees handle mixed categorical and numerical data, capture nonlinear interactions, are simple, fast learners. Trees also provide intrinsic feature selection scores through split values. The ACE feature selection algorithm [58] is summarized here. For a single decision tree the measure of variable importance is $VI(X_i, T) = \sum_{t \in T} \triangle I(X_i, t)$ where $\triangle I(X_i, t)$ is the impurity decrease due to an actual split on variable X_i at a node t of tree T. Impurity measure $I(t)$ for regression is defined as $\sum_{i \in t}(y_i - \bar{y})^2/N(t)$, where y_i is the response

of observation i in node t, and $\bar{y}$ is the average response for all $N(t)$ observations in node t. For classification, $I(t)$ equals the Gini index at node t

$$\text{Gini}(t) = \sum_{i \neq j} p_i^t p_j^t \tag{4.1}$$

where p_i^t is the proportion of observations with $y = i$ and i and j run through all target class values. The split weight measure $\triangle I(X_i, t)$ can be improved if out-of-bag (OOB) samples are used. The split value for the selected variable is calculated using the training data as usual. However, only the OOB samples are used to select the feature as the primary splitter. The experiments show that this provides a more accurate estimate of variable importance, and mitigates the cardinality problem of feature selection with trees [88] (where features with greater numbers of attributes values are scored higher by the usual metrics). Then, the importance score in a ensemble can be obtained by averaging over the trees

$$E(X_i) = \frac{1}{M} \sum_{m=1}^{M} VI(X_i, T_M) \tag{4.2}$$

Furthermore, a statistical criterion is determined through the use of artificial features (permutations of the actual features). Variable importance scores for actual features are compared to the distribution of scores obtained for the artificial features. Replicates of the ensembles are also used so that a statistical t-test can generate a p-value for the importance score of an actual feature. Further comments are provided below.

4.2.2. Statistical criteria for identifying relevant and redundant features

For deleting irrelevant or redundant features, a threshold is needed. Artificial contrasts can be used to construct and specify the threshold in an efficient way. Let the

number of variables be M. Denote the variables set as $S_X = \{X_j, j = 1, 2, ...K\}$. In each replicate $r, r = 1, 2, ...R$, artificial variables are generated as follows. For every variable X_j in S_X, a corresponding artificial variables Z_j^r is generated from randomly permutating values of X_j, let $S_Z^r = \{Z_j^r, j = 1, 2, ...K\}$. Then the new variables set can be denoted as $S_{X,Z}^r = \{S_X, S_Z^r\}$.

Consider relevant variables selection. Denote the importance score of $S_{X,Z}^r$ as $I_{X,Z}^r = \{I_X^r, I_Z^r\}$, where $I_X^r = \{I_{X_j}^r, j = 1, 2, ...M)\}$ and $I_Z^r = \{I_{Z_j^r}, j = 1, 2, ...K)\}$, $I_{X_j}^r$ and $I_{Z_j^r}$ are the importance scores of X_j and Z_j^r at the r^{th} replicate respectively. Denote $I_{X_j} = \{I_{X_j}^r, r = 1, 2, ...R\}$. Then $I_{X,Z}^r$ can be obtained by using relevant feature selection methods to $S_{X,Z}^r$. Denote I_α^r as the $1 - \alpha$ percentile value of I_Z^r and $I_\alpha = \{I_\alpha^r, r = 1, 2, ..., R\}$. For each variable X_j, a paired t-test compares I_{X_j} to I_α. A test that results in statistical significance, i.e., a suitably small p-value, identifies an important variable. Therefore, an important variable here need consistently score higher than the artificial variables over multiple replicates.

Consider redundancy elimination. Let M_{X_i,X_j}^r for $j = 1, 2, ...i - 1, i + 1, ..., K$ and M_{X_i,Z_j}^r for $j = 1, 2, ..., K$ denote the masking score of X_i over X_j, and over Z_j^r for replicate $S_{X,Z}^r$ respectively. Denote $M_{X_i,\alpha}^r$ as the $1 - \alpha$ percentile value of M_{X_i,Z_j}^r and $M_{X_i,\alpha} = \{M_{X_i,\alpha}^r, r = 1, 2, ..., R\}$. A paired t-test compares between M_{X_i,X_j}^r and $M_{X_i,\alpha}$. Variable X_j is masked by variable X_i if the test is significant.

4.2.3. Residuals for multiple iterations

A single iteration in Algorithm 1 can select a relevant and non-redundant feature set, but it may fail to detect some variables important but possibly weaker than a primary set. Thus more iterations are considered here. At the end of each iteration, a subset of features $\hat{\Phi}$ can be obtained. An ensemble model $g_T(\hat{\Phi})$ is built on $\hat{\Phi}$. Denote $\hat{T}$ as the out-of-bag (OOB) prediction of $g_T(\hat{\Phi})$. Then residual are calculated and form a new target. For a regression problem, the new target is simply formed by: $T = T - \hat{T}$. For a classification problem, residuals are calculated from a multi-class logistic regression procedure. Log-odds of class probabilities for each class are predicted (typically a gradient boosted tree [97]is used), and then pseudo-residuals are taken as residuals.

In a Bayesian network sometimes non-causal, but relevant variables, can also contribute to the target. Though the contribution from those non-causal but relevant variables could be small compare to causal related variables, ACE adds them into the feature set. Therefore, false alarm rates might be increased. The Bonferroni correction is a multiple-comparison correction used when several statistical tests are performed simultaneously. The Bonferroni correction is used here to reduce the false positive rate. For example, if the p-value of t-test in the previous sections is α, when there are N features the p-value is reduced to α/N.

4.3. Experiments

The work here focuses on continuous Bayesian networks but an example from a discrete network is added to illustrate ACE [58] easily generalizes – the discrete networks results are equally good. ACE and feature selection methods correlation-based feature selection (CFS) [89], SVM-RFE [57], and fast correlation-Based filter (FCBF) [90] are applied to learn the MB of the target nodes. The performance is also compared to a well-known Bayesian local structure learning algorithm: max-min parents and children (MMPC) [98]. In the experiments, ACE [58] is programmed in C, RWeka [92, 93] and bnlearn [99] in R [91] are used to run the other algorithms. The default parameter settings for the methods in the software are used. To evaluate the performance, the sensitivity and specificity are measured for a given task. The sensitivity is the ratio of the number of correctly identified variables in the MB over the size of the true MB. The specificity is the ratio of the number of correctly identified variables as not belonging in the MB over the true number of variables not in MB [98]. To compare different algorithms, a combined measure d is used [98].

$$d = \sqrt{(1 - \text{sensitivity})^2 + (1 - \text{specificity})^2} \tag{4.3}$$

A better algorithm implies a smaller d value.

4.3.1. Continuous, Gaussian local structure learning

There are few available continuous benchmark causal-structure network (the focus is on discrete networks). Therefore a causal-structure network with continuous

nodes is simulated as shown Figure 12. Bayesian structure learning often assumes Gaussian models whereas the ensemble-based ACE method is not limited to the such models. The first experiment uses the common Gaussian distributions for these experiments and the second experiment relaxes this assumption. Because FCBF and SVM-RFE (in RWeka [92, 93]) do not work with continuous target variables, only ACE, MMPC and CFS with best first search (CFSBestFirst) and gene search (CFSGene) are applied to this data.

Consider the network in Figure 12. For the first experiment nodes A, B, C are root nodes and follow normal distributions N(1,1), N(2,1) N(3,1), respectively, where $N(\mu, \sigma^2)$ denotes a normal distribution with mean μ and variance σ^2. Denote a node (not a root node) as N_i, and denote the parent nodes of N_i as $N_i^p(j), j = 1, ..., |N_i^p|$, where $|N_i^p|$ is the number of parent nodes of N_i. The causal relation between N_i and $N_i^p(j)$ is expressed by $N_i = f(N_i^p(j))$. Here $N_i = \sum_{j=1}^{|N_i^p|}(N_i^p(j)) + \varepsilon$ or $N_i = \prod_{j=1}^{|N_i^p|}(N_i^p(j)) + \varepsilon$ where $\varepsilon \sim N(0, 1)$. Therefore, both linear and nonlinear causal relationships in the network can be investigated. For example, in Figure 12, the linear causal relationship between node D and its parent nodes A, C is $D = A + C + \varepsilon$. The nonlinear causal relationship is $D = A * C + \varepsilon$. For each of the continuous Bayesian networks, 5000 rows of data are simulated. The objective is to learn the MBs of the output nodes.

The sensitivity, specificity and combined measure d for the linear and non-linear cases are shown in Table 10-12. For the linear Bayesian network, it is well known that linear relationships are not optimal for a tree representation. Still ACE has the lowest d

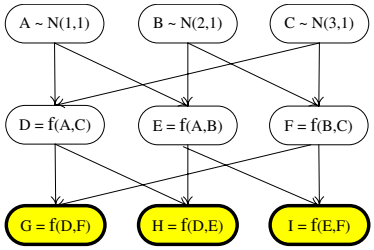


Figure 12. Nodes with thick edges (yellow) are taken as targets. The function f is taken as either an additive or multiplicative function of the inputs.

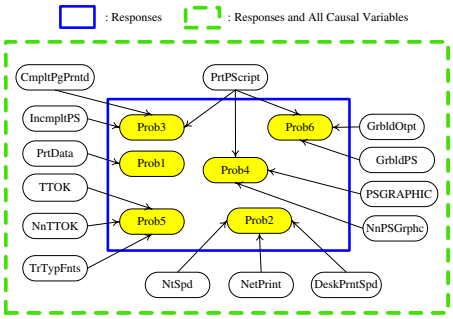


Figure 13. Local structure of the Windows printer network with regard to targets.

TABLE 10
Sensitivity for each output node from different algorithms learning continuous, Gaussian, linear and nonlinear Bayesian networks

	Linear				NonLinear			
	G	H	I	Average	G	H	I	Average
ACE	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
CFSBestFirst	1.00	1.00	1.00	1.00	1.00	1.00	0.50	0.83
CFSGene	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
MMPC	1.00	1.00	1.00	1.00	1.00	1.00	0.50	0.83

value. The other three methods have the same d value. For the non-linear network the challenge of learning increases, and the d of all methods increase. ACE still produces the smallest d value.

4.3.2. Continuous, nonGaussian local structure learning

For the nonGaussian experiment the distributions for nodes A, B, C are changed to $Normal(0, 1)$, $Exponential(1)$, $Uniform(-1, 1)$ respectively. Other characteristics of the experiment (including the linear and nonlinear target functions) are the same

TABLE 11
Specificity for each output node from different algorithms learning continuous, Gaussian, linear and nonlinear Bayesian networks

	Linear				NonLinear			
	G	H	I	Average	G	H	I	Average
ACE	1.00	1.00	0.83	0.94	1.00	0.67	1.00	0.89
CFSBestFirst	0.67	0.67	0.67	0.67	0.50	0.33	0.33	0.39
CFSGene	0.67	0.67	0.67	0.67	0.50	0.33	0.33	0.39
MMPC	0.67	0.67	0.67	0.67	0.00	0.67	0.17	0.28

TABLE 12
Combined measure d for each output node from different algorithms learning continuous, Gaussian, linear and nonlinear Bayesian networks

	Linear				NonLinear			
	G	H	I	Average	G	H	I	Average
ACE	0.00	0.00	0.17	0.06	0.00	0.33	0.00	0.11
CFSBestFirst	0.33	0.33	0.33	0.33	0.50	0.67	0.83	0.67
CFSGene	0.33	0.33	0.33	0.33	0.50	0.67	0.67	0.61
MMPC	0.33	0.33	0.33	0.33	1.00	0.33	0.97	0.77

TABLE 13
Sensitivity for each output node from different algorithms learning continuous, nonGaussian, linear and nonlinear Bayesian networks

	Linear				NonLinear			
	G	H	I	Average	G	H	I	Average
ACE	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
CFSBestFirst	1.00	1.00	1.00	1.00	0.00	0.00	0.50	0.17
CFSGene	1.00	1.00	1.00	1.00	0.00	0.00	0.50	0.17
MMPC	1.00	1.00	1.00	1.00	0.00	0.00	0.50	0.17

TABLE 14
Specificity for each output node from different algorithms learning continuous, nonGaussian, linear and nonlinear Bayesian networks

	Linear				NonLinear			
	G	H	I	Average	G	H	I	Average
ACE	1.00	1.00	0.67	0.89	0.50	0.50	0.33	0.44
CFSBestFirst	0.67	0.83	0.83	0.78	0.50	0.33	0.50	0.44
CFSGene	0.67	0.83	0.83	0.78	0.50	0.33	0.50	0.44
MMPC	0.50	0.50	0.67	0.56	0.50	0.33	0.50	0.44

as in the Gaussian case. The results are shown in Table 13-15.

For both nonGaussian linear and nonlinear networks, ACE is still better than the other three methods. CFSBestFirst outperforms MMPC in the nonGaussian linear case, while they have similar performance in other cases. Consequently, feature selection methods can provide reasonable alternatives to the MMPC algorithm in continuous networks. Furthermore, it is more difficult for all methods to learn a nonlinear relationship than a linear relationship in the nonGaussian cases.

TABLE 15
Combined measure d for each output node from different algorithms learning continuous, nonGaussian, linear and nonlinear Bayesian networks

	Linear				NonLinear			
	G	H	I	Average	G	H	I	Average
ACE	0.00	0.00	0.33	0.11	0.50	0.50	0.67	0.56
CFSBestFirst	0.33	0.17	0.17	0.22	1.12	1.20	0.71	1.01
CFSGene	0.33	0.17	0.17	0.22	1.12	1.20	0.71	1.01
MMPC	0.50	0.50	0.33	0.44	1.12	1.20	0.71	1.01

4.3.3. Discrete local structure learning

Although this work focuses continuous network structure, a discrete Bayesian network is also considered. The network is Windows printer trouble shooting with 76 features. 10000 observations were generated with the GeNIe structural modeling tool (http://genie.sis.pitt.edu/). The local structure with regard to the targets of the network are illustrated in Figure 13. Here 6 nodes (printer problem nodes) are considered as the targets (each with binary classes). ACE, MMPC, FCBF, CFSBestFirst, CFSGene, SVM-RFE are compared based on learning the local structure of the Bayesian networks. Because SVM-RFE requires the number of features to be selected as an input, the number of features is assigned in two ways: the size of the correct Markov blanket and the number of features selected by ACE. The SVM-RFEs with these two parameters are denoted as SVM(MB) and SVM(ACE), respectively. The results from the Windows printer trouble shooting network are shown in Table 16-18.

For the Windows printer network, ACE and SVM(MB) have the lowest d values.

TABLE 16
Sensitivity of outputs from different algorithms learning the Windows printer network. SVM(MB) is given the correct number of features, and SVM(ACE) is given the number of features selected by ACE

	Pro1	Prob2	Prob3	Prob4	Prob5	Prob6	Average
ACE	1.00	1.00	0.67	1.00	1.00	0.33	0.833
CFSBestFirst	1.00	1.00	0.67	1.00	0.67	0.67	0.833
CFSGene	1.00	1.00	0.67	0.67	1.00	1.00	0.889
FCBF	1.00	1.00	0.33	1.00	0.67	0.33	0.722
MMPC	1.00	1.00	0.33	0.67	1.00	0.33	0.722
SVM(ACE)	1.00	1.00	0.33	1.00	1.00	0.33	0.778
SVM(MB)	1.00	1.00	0.67	1.00	1.00	0.67	0.889

TABLE 17
Specificity of outputs from different algorithms learning the Windows printer network. SVM(MB) is given the correct number of features, and SVM(ACE) is given the number of features selected by ACE

	Pro1	Prob2	Prob3	Prob4	Prob5	Prob6	Average
ACE	1.00	1.00	1.00	1.00	1.00	1.00	1.000
CFSBestFirst	0.89	0.97	0.96	0.96	0.94	0.93	0.943
CFSGene	0.66	0.81	0.76	0.85	0.75	0.81	0.772
FCBF	1.00	0.97	1.00	0.96	0.93	1.00	0.977
MMPC	1.00	0.96	1.00	0.99	0.97	1.00	0.986
SVM(ACE)	1.00	1.00	0.99	1.00	1.00	1.00	0.998
SVM(MB)	1.00	1.00	0.99	1.00	1.00	0.99	0.995

TABLE 18
Combined measure: d of outputs from different algorithms learning the Windows printer network. SVM(MB) is given the correct number of features, and SVM(ACE) is given the number of features selected by ACE

	Pro1	Prob2	Prob3	Prob4	Prob5	Prob6	Average
ACE	0.00	0.00	0.33	0.00	0.00	0.67	0.167
CFSBestFirst	0.11	0.03	0.34	0.04	0.34	0.34	0.199
CFSGene	0.34	0.19	0.41	0.37	0.25	0.19	0.292
FCBF	0.00	0.03	0.67	0.04	0.34	0.67	0.291
MMPC	0.00	0.04	0.67	0.33	0.03	0.67	0.289
SVM(ACE)	0.00	0.00	0.67	0.00	0.00	0.67	0.222
SVM(MB)	0.00	0.00	0.33	0.00	0.00	0.33	0.111

SVM(MB) only outperforms ACE for the target Prob6. However, SVM(MB) is given the priori knowledge of the size of true MBs. With the number of variables selected by ACE as input, SVM(ACE) does not perform as well as ACE. Another feature selection method CFSBestFirst also provides better results than MMPC.

4.4. Conclusions

Structure learning is important for both discrete and continuous networks, and relaxed Gaussian assumptions are important for continuous networks. Common feature selection methods, along with a Bayesian structure algorithm, are compared for the structure learning problem, and experiments illustrates the strength of an ensemble-based feature selection approach in these cases.

CHAPTER 5

A TREE ENSEMBLE FOR TIME SERIES CLASSIFICATION AND FEATURE EXTRACTION

5.1. Introduction

Time series classification [10–12, 60–62] has been an active research topic in recent years. Time series classification methods can be divided into instance-based and feature-based methods. Instance-based classifiers (e.g. [60, 63]) predict a test instance based on its similarity to the training instances. Among instance-based classifiers, one-nearest-neighbor (1NN) classifiers with a Euclidean distance metric (NNEuclidean) or a dynamic time warping metric (NNDTW) have been widely, and successfully used [63–67]. Usually NNDTW performs better than NNEuclidean since DTW [13] is robust to the distortion of time axis, and is considered as a strong solution for time series problems [15]. A disadvantage of NNDTW is lack of interpretability. To this end, [68] proposed a decision tree approach, which split examples based on dynamic warping distance between a pair of time sequences. Such a decision tree improves interpretability over NNDTW to some degree, and speeds up the time for testing, but the interpretability is still limited.

Feature-based classifiers extract interpretable features, and then input the features to conventional classifiers, which generally are more interpretable than instance-based classifiers. [69] incorporated domain knowledge into an automated search for extracting features, and then applied conventional classifiers such as a support vector machine (SVM) [70]. [71] extracted statistical features such as mean and deviation from

time series, and then a multi-layer perceptron neural network was used for classification. [72] extracted features from intervals of time series, and SVM was trained on the features.

[73] extracted features from intervals of time series, and built boosting binary stumps, while [74] used the features from boosting binary stumps to build a single decision tree, which leads to a comprehensible classifier. However, only binary stumps were boosted, and the effect of using more complex base learners, such as decision trees, should be studied [73] (but larger tree models impact the computational complexity of their method). This work shows that this issue can be handled with a simple random strategy (linear in the length of the time series). More importantly, [73, 74] used only class-based information to evaluate the features. However, the number of extracted features is generally large, sometimes even larger than the number of training examples, and a number of features can provide the same class information (e.g., entropy gain). Consequently, class entropy alone can result in features useful for generalization being ignored.

Advances in feature selection [58], allowed [75] to consider a massive number of features. Features sets were derived from statistical moments, wavelets, Chebyshev coefficients, principle component analysis (PCA) coefficients, and the original values of the signal. Furthermore, time-warped invariant versions of these features were generated as described by [76]. This massive set of features were used in a dynamic gradient-boosted tree (GBT) ensemble [77] for classification. The dynamic enhancement allows the ensemble to provide higher weight to features with more potential and reduce the

computational burden. Performance was excellent in the time series classification challenge [75]. Still, it is computationally complex and features such as PCA scores and Chebyshev coefficients are difficult to interpret and the GBT classifier obfuscates results further. The objective of this work is to produce a fast, accurate classifier that yields a simple set of features that can contribute to the domain knowledge. For example, in manufacturing applications, specific properties of the signals that discriminate conforming from un-conforming product is invaluable to diagnose, correct, and improve processes.

Ensemble methods have been shown effective for time series classification [75, 78, 79]. However, lack of interpretability, and high computational complexity are disadvantages of the current ensemble methods. Here an effective, yet efficient, ensemble method, referred to as a time series forest (TSF) is proposed for time series classification and feature extraction: 1) with new criteria to evaluate features. The new criteria consider both class-based information and distance measures. Experimental studies show that the new criteria produce significantly better performance than only a class-based criterion. 2) with random sampling based on intervals in a method analogous to a random forest [88]. Consequently, model variance can be reduced and the computational complexity of TSF is only linear in the length of the time series: $O(MN \log N)$ where N is the number of time series for training, and M is the length of each time series). 3) and TSF can extract a scalable set of interpretable features.

The remainder of this paper is organized as follow. Section 5.2 describes the time series forest (TSF) method. Section 5.3 illustrates extracting time series features

via TSF. Section 5.4 demonstrates the effectiveness and efficiency of TSF by testing on a full set of benchmark data set from UCR time series database [100]. Conclusions are drawn in Section 5.5.

5.2. Time series forest classifier

This section describes the TSF algorithm. First introduce the split of a tree node, and then describe the algorithm for a tree. TSF is a combination of such trees. To build a tree classifier, the split at each node needs to be defined. A split consists of two important components: search space and evaluation criteria (also referred to as loss function). The search space contains all the candidate splits of interest, and the evaluation criteria defines the best split in the search space.

Assume the time series are measured at equally-spaced intervals, and each time series is of the same length M, but the methods can be modified for more general cases. A univariate time series is denoted as $\{v_1, v_2, \ldots, v_M\}$. Though this method can be easily extended for multivariate time series, here focuses on univariate time series benchmark data sets in the experiment, and thus adopt the univariate notation for simplicity. A time series data set consists of N time series $\{e_1, e_2, ..., e_N\}$. Each instance is associated with a class label y_n, for $n = 1, 2, \ldots, N$ and $y_n \in \{1, 2, ..., C\}$. Given a set of time series for training and the class labels for these instances, the goal is to predict the class labels for testing instances. The goal of feature extraction is to extract interpretable features informative for classification.

At each node a time interval is selected, and then features (such as mean, deviation, etc.) are extracted from that interval. The interval and the feature are chosen to maximize the splitting criterion when the collection of time series is partitioned into child nodes. A typical example of a interval splitting rule is average$(I[t_1, t_2]) \leq$ threshold, where the notation indicates that a series for which the average over the interval $[t_1, t_2]$ is less than or equal to a threshold is assigned to the left child, and to the right child otherwise. An interval tree is able to capture interpretable features from a window from time series, however, $O(M^2)$ calculations are needed to evaluate all the intervals, and thus, expensive for time series with large length. Previous research used sampling powers of two to reduce the computational complexity from $O(NM^2)$ to $O(NM \log M)$ [73] (at the root node) for univariate time series. An alternative is provided in the next section that reduce this to $O(NM)$. Furthermore, the powers of two can generate wide intervals (for longer time series) that can attenuate important features.

5.2.1. Search space

Let $f_k(\cdot)$ $(k = 1, 2, ..., K)$ denote the k^{th} feature type. Here consider features such as $f_1 = mean$, $f_2 = slope$, $f_3 = variance$. Let $f_k(t_1, t_2)$ for $(0 < t_1 \leq t_2 \leq M)$ denote an feature calculated over the interval between t_1 and t_2. A candidate split S in a tree node (for simplicity, assume the number of instances equals N, i.e., the split is the root node) tests the following condition

$$f_k(t_1, t_2) \leq \text{threshold} \tag{5.1}$$

The value of the threshold depends on the feature type and time interval. Given a time series instance e_n let $f_k^n(t_1, t_2)$ denote the interval feature calculate from instance n. The threshold set for $f_k(t_1, t_2)$ can be defined as $\{f_k^n(t_1, t_2), n \in 1, 2, ..., N\}$, i.e., the values of $f_k(t_1, t_2)$ over all time series instances. Typically decision trees order the threshold set and evaluate the information gain over all possible values in the threshold set. Here consider a more efficient approach [73]. The thresholds for $f_k(t_1, t_2)$ are formed such that they divide the range of $[\min(f_k^n(t_1, t_2)), \max(f_k^n(t_1, t_2)], n \in 1, 2, ..., N$ with equal width. The number of thresholds are fixed for all splits, e.g. 20, and denoted as |threshold|.

Therefore, the search space Θ at a tree node consists of a set of time intervals $\{t_1, t_2 \in 1, 2, ...M, t_1 \leq t_2\}$, a set of features $\{f_1, f_2, ..., f_K\}$, and a threshold set. The size of the time interval space is $M(1 + M)/2$. The size of space of feature space is K. The size of Θ is $|\Theta_t| * K * |\text{threshold}|$. The search space for splitting evaluation leads to computational complexity of $O(NM^2)$ at the node.

A random forest (RF) [88] is an ensemble of base tree models.At each node, a RF considers the best split based on only a random sample of feature subspaces. Often the sample size is $\sqrt{p}$, where p is the number of predictor features. The random feature selection reduces the variance of the ensemble [88], and also reduces the computational complexity of a single tree from $O(pN \log N)$ to $O(\sqrt{p}N \log N)$ (assume the depth of tree is $O(\log N)$ where N is the number of instances).

Similar to a RF the sampling procedure at each node for TSF considers only a fraction of the potential splits point. This yields the same benefits (reduced variance and

computations). The sampling is illustrated in Algorithm 2, and reduces the computational complexity to $O(NM)$ (at the root node).

Algorithm 2 $sample()$ function: randomly sampling a set of start position and end position $< T_1, T_2 >$ for interval features.

$< T_0, T_1 >= empty$
Randomly select a $\sqrt{M}$ number of window sizes from $\{1, ..., M\}$ without replacement. Denote the set of window sizes as W.
for w in set W **do**
  randomly select $\sqrt{M - w + 1}$ number of start positions from $\{1, ..., M - w + 1\}$ without replacement. Denote the set of start positions as T_0.
  for t_0 in set T_0 **do**
    $< T_0, T_1 > \leftarrow + < t_0, t_0 + w - 1 >$
  end for
end for
Output $< T_0, T_1 >$

5.2.2. Evaluation criteria

A split at a tree node tests equation (5.1) and divides the examples at a node into two children nodes. An evaluation criteria is needed to select the best split S^*: $f_*(p_1^*, p_2^*) <= threhold^*$ from the search space Θ. Information gain measures such as entropy gain are commonly used for such an evaluation in tree models such as C4.5 [95]. Denote the proportion of instances with the corresponding classes at the node as $\{\gamma_1, \gamma_2, ...\gamma_C\}$. The entropy at the node is defined as

$$Entropy = -\Sigma_{c=1}^{C} \gamma_c \log \gamma_c \tag{5.2}$$

The entropy gain $\triangle Entropy$ for a split is then the difference between the weighted sum of entropy at the children nodes and the entropy at the parent node, where the weight at

a child node is the proportion of the cases assigned to that child node.

$\triangle Entropy$ captures the class information gain, and can be used to evaluate the quality of a split shown in equation (5.1). However, in time series classification, the number of candidate splits could be larger than number of time series examples, and there are often cases that several candidate splits lead to the same entropy gain. Therefore alternative criteria are considered. One measure is called relative mean square error gain (RMSE). The mean square error of an interval feature $f_k(t_1, t_2)$ at a node is defined as

$$MSE = \frac{1}{\nu} * \Sigma_{n=1}^{\nu} (f_k^n(t_1, t_2) - \overline{f_k(t_1, t_2)})^2 \tag{5.3}$$

where $\overline{f_k(t_1, t_2)} = \Sigma_{n=1}^{\nu} f_k^n(t_1, t_2)/\nu$, and ν is the number instances at the node. Let MSE, MSE_l and MSE_r denote the MSE at the root, left, and right child nodes, respectively. The $RMSE$ gain due to splitting the node into two children nodes is

$$\triangle RMSE = 1 - \frac{MSE_l + MSE_r}{MSE} \tag{5.4}$$

Here $\triangle RMSE$ is used to evaluate the split from a clustering perspective. Larger $\triangle RMSE$ is preferred.

The second measure considers the margin between the threshold and its nearest neighbor

$$M = \min_{n=1,2,\ldots,\nu} |f_k^n(t_1, t_2) - \text{threshold}| \tag{5.5}$$

A relative margin (RM) can be defined as

$$RM = M / \max_{n,r=1,2,\ldots,\nu, n \neq r} [f_k^n(t_1, t_2) - f_k^r(t_1, t_2)] \tag{5.6}$$

A large relative margin is preferred for better generality and interpretability.

Finally, an evaluation criterion E for a split is defined as a linear combination

$$E = \triangle Entropy + \alpha RMSE + \beta RM \tag{5.7}$$

The split S^* with maximum E is selected. Small values for α and β are used so that the only role for the corresponding terms in the model is to break ties that can occur from entropy alone. Set $\beta = 0.000001 \ll \alpha = 0.001 \ll 1$. Though it may be possible to obtain more accurate predictions by adjusting α and β, the model becomes more complex.

5.2.3. Time series forest

Each tree in a TSF is briefly described in Algorithm 3. The data for each tree are randomly sampled (without replacement) with a specified percentage. Assume each tree has depth $O(\log N)$, the computational complexity for training a TSF is $O(MN \log N)$.

5.3. Feature extraction and summary

TSF selects the best interval feature $f_*(t_0^*, t_1^*)$ from a subspace at each node, and is therefore a feature selection process. Here discuss how to extract a scalable number of features, and how to summarize these features in an interpretable way.

An interval feature in TSF should be informative about predicting the class, and the entropy gain can be used to score the degree of information of the feature. However, because a large number of features could be extracted from TSF, and some of them are

Algorithm 3 $tree(data)$: A single tree function in TSF. The function $evaluate(f_k(t_0, t_1) \leq threshold, data)$ calculates the criterion in equation (5.7) on $data$.

if only one class exists in $data$ **then**
  terminal node;
  return;
end if
$\{T_0, T_1\}$=$sample()$
$E^* = 0$, ${t_0}^* = 0$,${t_1}^* = 0$, $threshold^* = 0$, $f_* = -1$
for $< t_0, t_1 >$ in set $< T_0, T_1 >$ **do**
  for $threshold$ in set $Threshold$ **do**
    for k in 1:K **do**
      $E = evaluate(f_k(t_0, t_1) \leq threshold, data)$
      if $E > E^*$ **then**
        $E^* = E$, ${t_0}^* = t_0$, ${t_1}^* = t_1$, $threshold^* = threshold$, $f_* = f_k$
      else
        if $E^* = 0$ **then**
          terminal node;
          return;
        end if
      end if
    end for
  end for
end for
$data_{left} \leftarrow$ time series with $f_*(t_0, t_1) \leq threshold, data)$
$data_{right} \leftarrow$ time series with $f_*(t_0, t_1) > threshold, data)$
$tree(data_{left})$
$tree(data_{right})$

redundant or ignorable, a further feature selection procedure is used. Feature subset selection methods such as correlation-based feature selection (CFS) [89], fast correlation-Based filter (FCBF) [101], artificial contrasts with ensembles (ACE) [58] are common choices. However, a large number of features from TSF could affect the efficiency of some of these feature subset selection methods. For example, the worse-case complexity of FCBF is $O(p^2)$, where p is the number of features. Consequently, redundant features

are eliminated by extracting features from only a small number of trees, and ignorable features are eliminated by discarding features appearing in tree nodes exceeding a certain depth.

Let the number of trees for feature extraction be $nTree$, and the maximum depth be $depth$, where the depth is the length of the path from the root node to the node. The number of features is bounded by $nTree * 2^{depth} - 1$. After limiting the number of features by setting a small number of trees, and maximum depth, feature selection methods can then be used on the features more efficiently.

The TSF classifier is much less interpretable than a simple classifier such as C4.5. A subset of features can be extracted, but how these features are summarized for classification is important as well. This work considers using a simple classifier to learn the features selected from the previous step.

5.4. Experiments

TSF is implemented in Matlab and C++. Multi-core computing is used for training the models, but only use single core for testing (because testing is fast). The experiments were run on Vista system with 6GB RAM, quad CPU (2.5GHz). The parameters are set as follows: number of trees = 200, $f(\cdot) = \{mean, slope, variance\}$, and $arctan$ is used for slope. Here 20 thresholds are used and $\alpha = 0.001$, $\beta = 0.000001$. The minimum time interval is constrained to 5 points for mean and slope, and 15 points for variance, for better interpretability. The largest window (i.e., the interval equal to the

entire time series) is always a candidate for splitting a node. For feature extraction, 50 trees are used, and the maximum depth is 5. TSF is tested on a full set of time series data sets from [100]. The data sets characteristics are shown in Table 19. The training/testing setting is the same as in [100]. The classification accuracy and classification interpretability are discussed in the following sections.

	number classes	training cases	testing cases	time series length
50words	50	450	455	270
Adiac	37	390	391	176
Beef	5	30	30	470
CBF	3	30	900	128
Coffee	2	28	28	286
ECG200	2	100	100	96
FaceAll	14	560	1,690	131
FaceFour	4	24	88	350
fish	7	175	175	463
GunPoint	2	50	150	150
Lighting2	2	60	61	637
Lighting7	7	70	73	319
OliveOil	4	30	30	570
OSULeaf	6	200	242	427
SwedishLeaf	15	500	625	128
syntheticcontrol	6	300	300	60
Trace	4	100	100	275
TwoPatterns	4	1,000	4,000	128
wafer	2	1,000	6,174	152
yoga	2	300	3000	426

TABLE 19
Characteristics of the time series: number of classes, number of training and testing instances, and lengths of time series.

5.4.1. Classification accuracy and speed

Now evaluate the effectiveness of the evaluation criteria in equation 5.7, and the random sampling in Algorithm 2. The classification error rates and the computational time are used for evaluation. The results are shown in Table 20. In the table, "default" represents TSF with the proposed settings. The other entries in the table use the same as settings as default with the following exceptions: "entropy" uses only $\triangle Entropy$ for evaluating splits, "full sampling" searches all the intervals in the search space, "log sampling" searches window sizes that are powers of 2, "log+entropy" uses only $\triangle Entropy$ and searches window sizes that are power of 2. Because both "default" and "entropy" use random sampling, their times are similar; similarly "log sampling" and "log+entropy" have similar times, and thus only the times of the first ones are shown in the table. The testing times for all methods are small, and only the times of "default" are shown.

The mean and median are calculated for error rates and computation time for all data sets. Pair t-tests are performed between error rates of "default" and other methods. It can be seen that "default" is more accurate than both "entropy" and "log+entropy" at the significance level of 0.05. Consequently, the evaluation criteria in equation 5.7 improves the classifier's accuracy. There are no significant difference of classification accuracy between different sampling methods. However, the time for "default" is obviously smaller than "full sampling", "log sampling", and "log+entropy". In addition, the testing times for TSF are small, which can be a computing advantage over lazy learners

such as nearest neighbor classifiers.

	Error rates					training time			testing time
	default	entropy	log +entropy	full sampling	log sampling	default	full sampling	log sampling	default
50words	0.266	0.299	0.295	0.275	0.277	251.52	38150.00	1182.30	4.60
Adiac	0.263	0.279	0.274	0.261	0.215	139.57	12193.00	549.48	3.21
Beef	0.233	0.400	0.300	0.300	0.167	15.44	3487.20	74.98	0.26
CBF	0.043	0.068	0.140	0.034	0.020	2.44	114.54	6.53	0.66
Coffee	0.036	0.036	0.143	0.036	0.000	2.41	306.46	9.48	0.06
ECG200	0.000	0.010	0.010	0.000	0.000	1.50	61.64	4.34	0.07
FaceAll	0.237	0.246	0.253	0.148	0.224	85.54	5306.90	298.25	6.33
FaceFour	0.046	0.227	0.227	0.034	0.034	6.71	1217.60	30.98	0.27
fish	0.154	0.160	0.177	0.160	0.171	81.69	20244.00	409.86	1.52
GunPoint	0.073	0.100	0.087	0.067	0.073	3.57	206.57	9.75	0.17
Lighting2	0.197	0.279	0.295	0.262	0.246	25.78	6684.60	109.54	0.50
Lighting7	0.260	0.260	0.301	0.329	0.343	27.95	4072.00	110.71	0.51
OliveOil	0.133	0.100	0.167	0.100	0.100	10.89	3687.10	64.65	0.15
OSULeaf	0.422	0.413	0.409	0.401	0.393	108.69	25540.00	576.91	2.59
SwedishLeaf	0.136	0.131	0.176	0.149	0.142	82.23	5080.30	283.77	2.70
syntheticcontrol	0.023	0.027	0.040	0.043	0.043	9.65	231.31	20.93	0.45
Trace	0.000	0.010	0.020	0.010	0.000	9.55	1539.70	45.46	0.20
TwoPatterns	0.056	0.059	0.061	0.065	0.060	107.14	6613.00	396.43	17.72
wafer	0.007	0.008	0.010	0.011	0.009	37.84	2609.10	117.16	13.39
yoga	0.160	0.164	0.182	0.173	0.175	116.14	24244.00	531.63	45.33
mean	0.1373	0.1637	0.1783	0.1428	0.1346	56.312	8079.451	241.657	5.035
median	0.1347	0.1456	0.1766	0.1240	0.1212	26.868	3879.550	110.125	0.585
t-test	-	0.0451	0.0010	0.4904	0.7216	-	-	-	-

TABLE 20

Error rates and running time (seconds) for time series forest (TSF) with default and alternative settings. Multi-core computing is used to train models on a four-core computer. Only one core is used for testing. Here "default" represents TSF with the proposed settings. The other entries in the table use the same as settings as default with the following exceptions: "entropy" uses only $\triangle Entropy$ for evaluating splits, "full sampling" searches over all intervals, "log sampling" searches window sizes that are powers of 2, "log+entropy" uses only $\triangle Entropy$ and searches window sizes that are powers of 2.

Now compare TSF with the default settings to other classifiers: random forest with 500 trees used on the original time values, nearest neighbors classifier (NN) with Euclidean distance, nearest neighbors (NN) with dynamic time warping (DTW), and a GBT method with a massive number of features (Massive) [75]. This work considers

two versions of NN with DTW, "NNDTWBest" [65] searches the best warping window, while "NNDTWNoWin" has no warping window. The results for the three NN classifiers are obtained from [100]. Mean and median of error rates from all data sets are calculated, and paired t-tests between TSF and all other classifiers are performed. It can be seen from Table 21 that TSF has an obvious advantage over NN Euclidean and Random Forest and it is significantly better than NNDTWNoWin at significance level of 0.10. TSF is not significantly different from "NNDTWBest", but the mean and median error rates of TSF are obviously smaller. The Massive classifier is a strong performer from the time series challenge [75], with results better than TSF, but as mentioned in Section 5.1, the results are not interpretable and the method is computationally complex. Note that DTW is a strong solution for time series problems in a variety of domains [15].

The robustness of TSF accuracy to the number of trees is investigated. Figure 14 shows the change of average error rate from all data sets as the number of trees increases. The error rate decreases as the number of trees increases, but the change is relatively small after the number of trees achieves 100.

5.4.2. Feature extraction and summary

Features are extracted from TSF built from the training instances. To reduce the number of features, set the number of trees for feature extraction to 50, and the maximum depth as 5. Therefore, the maximum number of features that can be extracted is $50 * 2^5 - 1 = 1599$. In addition, CFS [89] and ACE [58] is used to select a subset

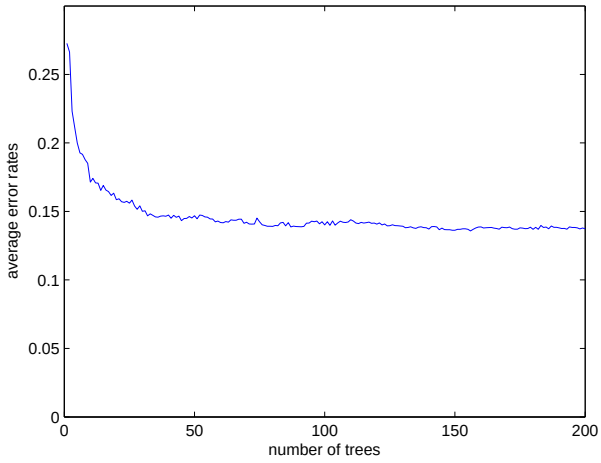


Figure 14. The average error rates from all data sets decreases as the number of trees increases. The error rates become stable when the number of trees is more than 100.

of features. ACE is tuned for two-class data, and therefore is only applied to two-class data sets. To summarize these features for classification, input the selected subset into interpretable classifiers: C4.5 [95] and naive Bayes (NB). Then the error rates for these classifiers are evaluated.

The results are shown in Table 22. The length of the time series was shown in Table , denoted as M. The initial number of features extracted from TSF is denoted as initTSF. The sizes of the subsets from feature selection are denoted as nCFS and nACE. The error rates of C4.5 and NB using the original time series values (C4.5 and NB), TSF interval features (C4.5TSF, NBTSF), and results after feature selection (C4.5CFS, C4.5ACE, NBCFS, NBACE) are shown in the remainder of Table 22.

The mean and median of error rates and number of features from all data sets are calculated. Paired t-tests between a classifier with CFS and a classifier with other

feature sets are performed. After feature selection, nCFS is much smaller than initTSF, which improves interpretability. By using interval features TSF, CFS, and ACE, for both C4.5 and NB, are better than using the original time values at significance level of 0.001. Furthermore, the accuracy of both classifiers using selected features from CFS is at least as good as the classifiers using TSF. The accuracy of C4.5 using selected features from ACE is competitive to C4.5 with other feature sets for data sets with two classes.

	TSF default	NN Euclidean	Random Forest	NNDTW Best	NNDTW NoWin	Massive
50words	0.266	0.369	0.332	0.242	0.310	0.235
Adiac	0.263	0.389	0.350	0.391	0.396	0.274
Beef	0.233	0.467	0.467	0.467	0.500	0.130
CBF	0.043	0.148	0.116	0.004	0.003	0.019
Coffee	0.036	0.250	0.321	0.179	0.179	0.004
ECG200	0.000	0.120	0.190	0.120	0.230	0.052
FaceAll	0.237	0.286	0.184	0.192	0.192	0.191
FaceFour	0.046	0.216	0.193	0.114	0.170	0.056
fish	0.154	0.217	0.211	0.160	0.167	0.147
GunPoint	0.073	0.087	0.093	0.087	0.093	0.079
Lighting2	0.197	0.246	0.246	0.131	0.131	0.131
Lighting7	0.260	0.425	0.247	0.288	0.274	0.256
OliveOil	0.133	0.133	0.200	0.167	0.133	0.170
OSULeaf	0.422	0.483	0.496	0.384	0.409	0.355
SwedishLeaf	0.136	0.213	0.117	0.157	0.210	0.107
syntheticcontrol	0.023	0.120	0.040	0.017	0.007	0.012
Trace	0.000	0.240	0.170	0.010	0.000	0.000
TwoPatterns	0.056	0.090	0.149	0.002	0.000	0.000
wafer	0.007	0.005	0.011	0.005	0.020	0.004
yoga	0.160	0.170	0.188	0.155	0.164	0.163
mean	0.1373	0.2337	0.2161	0.1636	0.1794	0.1192
median	0.1347	0.2165	0.1916	0.1560	0.1685	0.1185
t-test	-	0.0000	0.0007	0.1428	0.0556	0.0419

TABLE 21

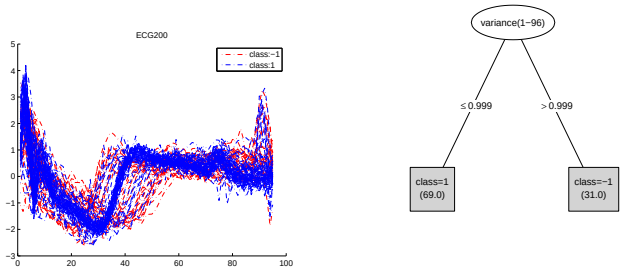
Time series forests (TSF) compared to alternatives: nearest neighbors (NN) based on Euclidean distance, random forest (500 trees), NN with dynamic time warping distance, a complex approach with a non-interpretable features (Massive). NNDTWBestWin searches for the best warping window, but NNDTWNoWin has no warping window. Paired t-tests between TSF and all other classifiers are performed

	Number of features				Error rate from C4.5				Error rate from NB			
	M	initTSF	nCFS	nACE	C4.5	C4.5TSF	C4.5CFS	C4.5ACE	NB	NBTSF	NBCFS	NBACE
50words	270	1487	78	-	0.585	0.503	0.503	-	0.437	0.387	0.404	-
Adiac	176	1292	67	-	0.458	0.402	0.427	-	0.437	0.286	0.246	-
Beef	470	270	9	-	0.433	0.633	0.400	-	0.500	0.367	0.300	-
CBF	128	40	11	-	0.327	0.192	0.192	-	0.104	0.032	0.018	-
Coffee	286	31	8	2	0.429	0.143	0.143	0.143	0.321	0.036	0.036	0.071
ECG200	96	1	1	1	0.280	0.020	0.020	0.010	0.230	0.000	0.000	0.000
FaceAll	131	1125	195	-	0.451	0.302	0.370	-	0.307	0.272	0.239	-
FaceFour	350	120	15	-	0.284	0.341	0.330	-	0.159	0.034	0.068	-
fish	463	762	79	-	0.389	0.280	0.280	-	0.331	0.240	0.206	-
GunPoint	150	67	5	6	0.227	0.147	0.180	0.147	0.213	0.260	0.213	0.313
Lighting2	637	220	14	-	0.377	0.180	0.230	-	0.328	0.311	0.213	-
Lighting7	319	473	32	12	0.452	0.370	0.370	0.230	0.356	0.247	0.288	0.344
OliveOil	570	134	22	-	0.233	0.067	0.200	-	0.233	0.067	0.067	-
OSULeaf	427	931	56	-	0.632	0.583	0.587	-	0.628	0.483	0.492	-
SwedishLeaf	128	1088	102	-	0.341	0.266	0.251	-	0.142	0.182	0.139	-
syntheticcontrol	60	242	51	-	0.190	0.057	0.060	-	0.040	0.020	0.013	-
Trace	275	117	3	-	0.210	0.050	0.070	-	0.200	0.000	0.020	-
TwoPatterns	128	847	56	-	0.349	0.127	0.125	-	0.543	0.204	0.127	-
wafer	152	148	8	11	0.018	0.012	0.008	0.008	0.292	0.010	0.009	0.023
yoga	426	525	31	21	0.316	0.254	0.258	0.257	0.458	0.333	0.340	0.335
mean	282.1	496.0	42.2	-	0.3490	0.2463	0.2502	-	0.3131	0.1885	0.1718	-
median	272.5	256.0	26.5	-	0.3448	0.2229	0.2404	-	0.3143	0.2218	0.1725	-
mean(2-class)	219.0	107.5	8.0	8.5	0.2978	0.1448	0.1614	0.1448	0.3065	0.1411	0.1245	0.1924
median(2-class)	238.2	207.5	14.2	8.8	0.2868	0.1575	0.1631	0.1323	0.3117	0.1476	0.1475	0.1811
t-test	0.000	0.000	-	-	0.0000	0.7975	-	-	0.0000	0.0588	-	-

TABLE 22

The length of the time series is denoted as M. The initial number of features extracted from TSF is denoted as initTSF. The subset sizes from feature selection are denoted as nCFS and nACE. The error rates of C4.5 and a naive Bayes (NB) classifier using the original time series values and the results after feature selection are shown in the remainder of the table. The mean, median of all data sets, and data sets with two-classes are provided. Paired t-tests compare feature sets from CFS to other feature subsets with the same classifier.

Figure 15 - Figure 18 show the decision tree models using the selected features from CFS as inputs for data sets ECG, Gunpoint, Trace and Wafer. These decision tree models capture the time series characteristics well, and demonstrate simple interpretability. It is also interesting that the decision tree model for ECG data set shown in Figure 15 indicates that the variance for the whole time series is a key feature to distinguish the time series with two classes. A further look into the ECG data set discloses that the variance is identically one for one class, and is less than one for the other class.

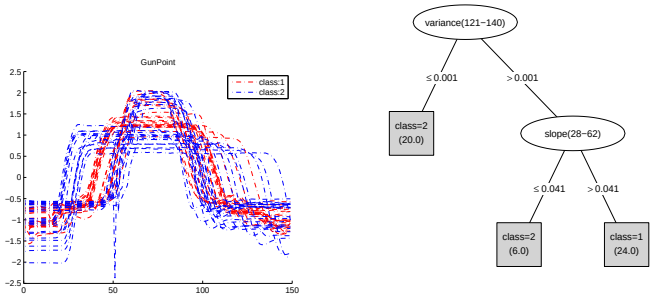


(a) Time series from the ECG data set (b) The tree classifier for the ECG data set

Figure 15. ECG time series and the corresponding decision tree classifier. The features used in the tree models are extracted from CFS feature selection applied to features used in TSF.

5.5. Conclusions

A time series forest (TSF) for time series classification, and feature extraction is proposed. TSF has computational complexity linear in the length of a time series, but is as accurate as a widely used alternative such as one nearest neighbor with the

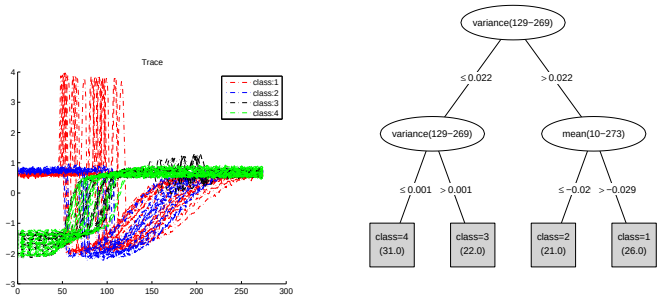


(a) Time series from the gunpoint data set

(b) Tree classifier for the gunpoint data set

Figure 16. Gunpoint time series and the corresponding decision tree classifier. The features used in the tree models are extracted from CFS feature selection applied to features used in TSF.

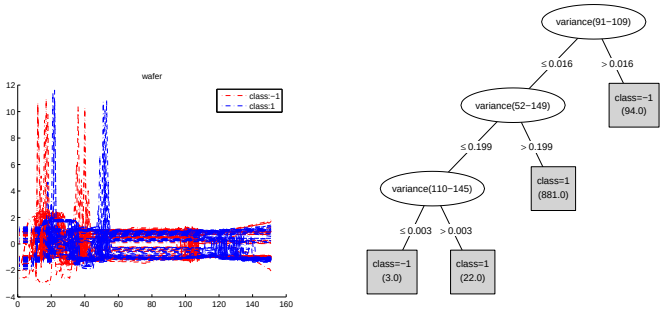
dynamic time warping metric. Furthermore, a scalable size of features can be extracted from TSF, and can be further reduced and summarized for better interpretability. The effectiveness, and efficiency are demonstrated by testing on a full set of benchmark data sets from UCR time series database [100]. The benchmark data sets are univariate, but the method can be also used for multivariate time series classification. One disadvantage of TSF is that it requires the time series to be aligned to the same length. Also TSF is not suitable for time series that have serious distortion on the time axis. As DTW is robust to the distortion of time axis, it would be worthwhile to integrate DTW into TSF without loss of much interpretability.



(a) Time series from trace data set

(b) The tree classifier for the trace data set

Figure 17. Trace time series and the corresponding decision tree classifier. The features used in the tree models are extracted from CFS feature selection applied to features used in TSF.



(a) Time series from the wafer data set

(b) The tree classifier for the wafer data set

Figure 18. Wafer time series and the corresponding decision tree classifier. The features used in the tree models are extracted from CFS feature selection applied to features used in TSF.

CHAPTER 6

BIAS OF IMPORTANCE MEASURES FOR MULTI-VALUED ATTRIBUTES AND SOLUTIONS

6.1. Introduction

Attribute importance measures for supervised learning are closely related to attribute selection and they are important for improving both learning accuracy and interpretability [16, 17]. There are well known attribute importance measures such as information-based measures. However, the bias problem for multi-valued attributes has been recognized for these methods. The number of distinct values of a attributes is referred to as its cardinality. [80] noted that attribute selection with Gini gain measure is biased in favor of those attributes with higher cardinality. [18] showed that there are biases in information-based measures adopted by decision tree inductions. [81] showed that attribute selection biases not only exist in information gain measures such as the Gini index, but also in others such as the distance measure in Relief [82], etc.

For solving the multi-valued problem, [83] introduced a normalization into the attribute selection measure known as the gain ratio. However, attributes with very low information values then appeared to receive an unfair advantage [18, 81]. Also [18] experimented with discrete, uniformly distributed attributes with different number of levels. They concluded that Chi-square could be used for the multi-valued problem. [81] proposed a minimum description length principle to alleviate the feature selection bias, but also mentioned that there are still slight decreases in the importance measure with the increasing cardinality.

Recently, a conditional inference framework [84] was proposed to solve the overfitting and attribute selection bias problems. [85] showed that this method demonstrated promising results in both null and power cases. In the null case, all predictor attributes are irrelevant with different cardinality. In the power case, only one predictor attribute is informative, all other attributes are irrelevant with different cardinality [85]. Though these research methods have successfully discovered and alleviated the multi-valued problem to some degree, there are still some important problems that are unresolved.

A permutation importance measure (PIMP) was introduced by [16]. It permutes the target attribute and a p-value can be used to measure importance. However, PIMP fits the importance score with a prior probability distribution. Though specifying a prior distribution is not necessary, [16] used prior probability distribution in their experiments (e.g., a gamma distribution for the simulated data, and a normal or lognormal distribution in other cases) and this requires such a prior to be specified in practice. One of the algorithms proposed here, pForest, also uses permutation importance, however, there are significant differences between PIMP and pForest. pForest permutes the predictors, and more importantly, makes use of a partial permutation strategy for better efficiency. Furthermore, pForest does not need prior probability distributions to be specified.

Most experiments from the existing research are limited to some idealized situations. For example, [18] considered the multi-valued problem only for irrelevant attributes, while [85] considered irrelevant attributes and only one informative attribute. [81] considered irrelevant and equally informative attributes. However, there can exist

both irrelevant and unequally informative attributes with different cardinalities. Furthermore, the informative attributes may interact with each other.

Therefore, it is important to consider the multi-valued problem under more realistic scenarios. Two two solutions are proposed for these problems. This work focuses on two-class classification problems (one of the most common problems in supervised learning), but it can be extended. This work also focuses on tree-based ensembles because of their capability to generate robust models that can handle nonlinearities, interactions, mixed (categorical and numerical) attributes, missing values, attribute scale differences, etc. However, the second method considered here is not limited to a certain type of classifier. It is a meta approach that can be applied to a base classifier to improve feature selection algorithms. Furthermore, this work contributes a more comprehensive simulation framework for studying the problem that integrates multiple cardinalities, and where non-equally informative attributes (with or without interactions) and irrelevant attributes co-exist. Such a framework can provide a useful benchmark to compare alternatives.

Section 6.2 briefly summarizes some widely used importance measures. Section 6.3 proposes two attribute importance methods. Section 6.4 describes the simulation framework and experimental results, while Section 6.5 provides conclusions.

6.2. Attribute importance measures

Several attribute importance measures are considered here. Random forest (RF) [88] is a commonly-used feature selection tool, e.g. [58, 102]. It allows for not only nonlinear models, but also interactions between predictor attributes. However, it can suffer from the multi-valued problem because it is based on an information criteria. Consequently, a remedy for RF's problem is important.

A RF builds an ensemble of decision trees. Each tree is built on a bootstrap sample (random, with replacement) from the original training data. Also, at each node only a subset of attributes is selected from the full set of attributes and the split is calculated only from members of this subset. The objective is to decrease the correlation between trees in the ensemble in order to decrease the final model variance. RF uses the Gini impurity criterion for scoring attribute importance. Denote $Imp(X_k, \tau)$ as the importance of a attribute X_i at a single tree τ, then $Imp(X_k, \tau) = \sum_{t \in \tau} \triangle Gini(X_k, t)$ where $\triangle Gini(X_k, t)$ is the Gini impurity decrease at a node t where X_k is the splitting attribute. The Gini index at node t is defined as $Gini(t) = \sum_j p_j^t(1 - p_j^t)$ where p_j^t is the proportions of cases of class j at node t. The importance of X_k is obtained from the sum of the importance scores from trees $\tau_m, m = 1, \ldots, M$ in a RF. For every tree τ in the ensemble, the instances not selected in the bootstrap sample are referred to as out of bag (OOB) and these cases can be considered to be a test sample for tree τ. These samples are used in the proposed importance measure.

A conditional inference framework [84] was proposed to solve the overfitting

problem and attribute selection bias problem. [85] used the method (referred as cForest) to measure importance for multi-valued attributes in a model similar to a RF. In this method, for each node, first the attribute to be split is selected by minimizing the statistical p value of a conditional inference independence test. Then the splitting value is established by an appropriate splitting criterion. The separation of attribute selection and splitting criterion is the key to handle the cardinality bias [84].

6.3. Attribute importance from OOBForest and pForest

This section proposes two methods to score attribute importance. The first method improves upon the RF methodology, while the second method can be applied to RF, but also more generally to other feature selection algorithms.

An OOBForest [58] is applied to determine importance measures. An OOBForest uses the training samples to find the best splitting value on each attribute in the same manner as for a RF (with the Gini index as the default information measure). But, instead of discarding the OOB samples when building a tree, the OOB samples are used to select the best splitting attribute at a node. That is, the Gini index (as the default) is recomputed for the OOB samples based on the split value obtained from the training data at each node. Furthermore, the importance measure $\triangle Imp(X_i, t)$ uses only the OOB samples. The principle here is similar to a conditional inference framework. The attribute selection criterion and splitting criterion are separated. The role of OOB samples was discussed for model improvements in [58], this work proposes to use it to

specifically solve the bias problem in measuring attribute importance. Computationally, the extra work over a RF is to calculate the split score from the OOB samples at each node in the forest. Because the OOB samples for a tree are typically smaller than the original training data less time is needed (approximately 2/3 less) than to generate a second RF (and the basic RF algorithm is fast [88]).

Next consider the pForest. Denote $X_k, k = 1, ..., K$ as the predictors and T as the target. [103] used permutation tests to obtain the statistical p value for dependency between an X_k and T. Then the inverse of the p value was used as the importance of the attribute. However, this method only measures the dependency of T over a single attribute X_k and the interactions between predictors are not considered. Permutation tests for feature selection was also used by [58]. Their method first randomly permutated each attribute $X_k, k = 1, ..., K$ and then compared importance score of an attribute to the distribution of scores from the irrelevant variables obtained form the permutations to obtain the corresponding attributes $Z_k, k = 1, ..., K$.

The proposed algorithm also uses permutations, but an attribute is only compared to permuted version of itself. Furthermore, the concept of partial permutations is introduced. In each replicate r, by applying an importance method $f(.)$ (such as RF) to $\{X_k, Z_k, T, k = 1, ..., K\}$, the importance score of $X_k, Z_k, k = 1, ..., K$, that is, $Imp_r(X_k)$ and $Imp_r(Z_k)$ can be obtained. A feature X_k is compared directly to its permuted version Z_k in each replicate to match the cardinality between X_k and Z_k (and

this differs from [58]). Next consider the measure

$$Imp(X_k) = \frac{1}{R}\sum_{r=1}^{R} I[Imp_r(X_k) > Imp_r(Z_k)] \tag{6.1}$$

where $I(.)$ denotes the indicator function. It can be seen that $\sum_{r=1}^{R} I(Imp_r(X_k) > Imp_r(Z_k))$ follows a binomial distribution $B(R, p_k)$, where p_k is the probability that $Imp(X_k) > Imp(Z_k)$. It is not feasible to compute the true p_k over all possible permutations in most practical situations. Therefore, [103] suggested a bounded number of permutations to achieve a significance level of 0.05.

The basic approach described so far is effective to distinguish informative from noninformative attributes. However, to rank informative attributes a more subtle refinement is used. For an informative attribute X_k one expects $Imp_r(X_k) > Imp_r(Z_k)$ in most replicates. In order to better detect finer importance relationships, partial permutations is proposed. That is, Z_k is obtained from permuting a fraction of the rows of X_k (a fraction δ selected randomly in each replicate). Consequently, as δ is decreased X_k and Z_k are more similar and it is more difficult for $Imp_r(X_k) > Imp_r(Z_k)$. Thus, only more informative attributes can achieve larger $Imp(X_k)$ values for small δ. The default choice is $\delta = 20\%$. This partial permutation method to attribute importance, with importance scores obtained from a RF, is referred to as the pForest.

Computationally, pForest is more demanding than OOBForest because each replicate requires another RF to be generated. However, the speed of a RF enables even hundreds of replicates to be computed in minutes for moderate data sets. Finally, note that although this work focuses on decision-tree ensembles, the permutation strategy

to solve the multi-valued problem can be applied to any feature selection method $f(.)$. One would simply replace the score $Imp_r(X_k)$ with another method and still average $I[Imp_r(X_k) > Imp_r(Z_k)]$ over the replicates.

Algorithm 4 pForest importance measure.

Input: R = number of permutation replicates; δ = percentage of rows permuted; training data $D = \{(x_i, t_i)|i = 1, \ldots, N\}$ with K features $F = \{X_1 ... X_K\}$; $f(F, D)$: a function that provides variables importance scores for attributes in F with data D (default is RF).

for r in 1:R **do**
  for k in 1:K **do**
    $Z_k \leftarrow$ randomly select and permute $\delta * N$ rows of X_k
  end for
  set $F' \leftarrow F \cup \{Z_1, \ldots, Z_K\}$
  $Imp_r(F') = f(F', D)$
end for
for k in 1:K **do**
  $Imp(X_k) = \frac{1}{R} \sum_{r=1}^{R} I(Imp_r(X_k) > Imp_r(Z_k))$
end for

Output: $Imp(X_k)$, for $k = 1...K$

6.4. Experiments

Similar to [18, 81, 85], The experiments are setup as simulations so that the "ground truths" for variable importance are known. The relationship between the predictor variables and the target variable is shown in Figure 19. Here T_1 and T_2 are the target attributes with and without interactions present in the model, respectively. All other variables are predictor variables. The generation and properties for these variables are summarized as follows:

- Generate $X_1 \sim Normal(0, 10)$, X_1 is then discretized (equal-frequency) into X_2 with 64 levels, X_3 with 16 levels, X_4 with 2 levels, X_5 with 8 levels, X_6 with 32 levels, respectively. Randomly permutate 30% of the rows of X_5, and 50% of the rows of X_6. This introduces different amount of noise into the X_5, X_6 so that they are unequally informative concerning the target.

- Generate $Y_k, k = 1, ..., 6$ independent from $X_k, k = 1, ..., 6$. The generation procedure is similar to the generation of $X_k, k = 1, ..., 6$.

- Generate $U_1 \sim Uniform(-10, 10)$, and U_1 is then discretized (equal-frequency) into U_2, U_3, U_4, U_5, U_6 with different cardinalities.

- The binary target T_1 is generated as $P(T_1 = X_4) = 0.95$ and $P(T_1 \neq X_4) = 0.05$.

- The binary target T_2 is generated as $P(T_2 = xor(X_4, Y_4)) = 0.95$ and $P(T_2 \neq xor(X_4, Y_4)) = 0.05$ (where xor is the exclusive or logical operation).

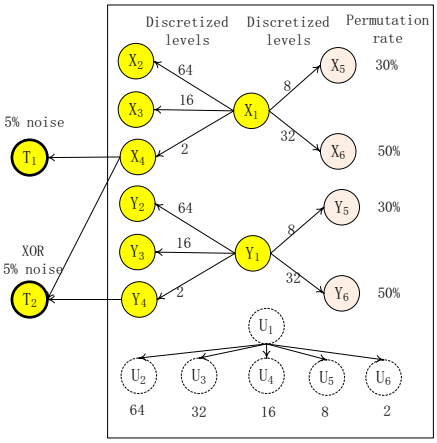


Figure 19. Relationship between predictors and targets along with cardinalities. Here T_1 and T_2 denote the target for the experiments with and without interactions, respectively.

Two experiments can be derived from the relationships among the attribute. In first experiment, T_1 is the target variable and $\{X_k, U_k, k = 1, ..., 6\}$ are the predictor variables. In the second experiment, T_2 is the target variable and $\{X_k, Y_k, U_k, k = 1, ..., 6\}$ are the predictor variables. The difference between the two experiments is that the true model for T_2 includes interactions from the xor function. In each experiment, 50 replicates of data sets are simulated, each data set with $5120 = 10 * 2^9$ rows of data so that all values of an attribute have the same number of rows. For example, for a two-value attribute, value 0 and value 1 each have 2560 rows.

By designing such experiments, the order of the importance scores for the predictor attributes is known. In the first experiment

$$\begin{aligned} Imp(X_1) &= \ldots = Imp(X_4) \\ &> Imp(X_5) \\ &> Imp(X_6) \\ &> Imp(U_1) = \ldots = Imp(U_6) \end{aligned}$$

Therefore, there are four groups and attributes from the same group have equal information regarding T_1.

In the second experiment

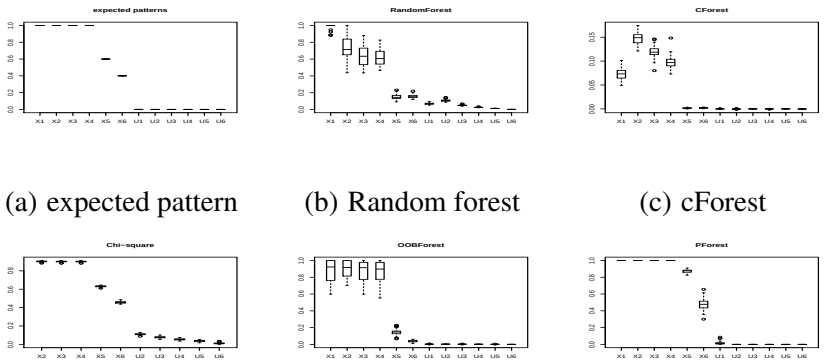
$$\begin{aligned} Imp(X_1) &= \ldots = Imp(X_4) = Imp(Y_1) = \ldots = Imp(Y_4) \\ &> Imp(X_5) = Imp(Y_5) \\ &> Imp(X_6) = Imp(Y_6) \\ &> Imp(U_1) = \ldots = Imp(U_6) \end{aligned}$$

Therefore, there are still four groups and attributes from the same group have equal information regarding T_2. A variable importance measure should be able to indicate such orders of variable importance. The original random forest [88], Chi-square, OOBForest [58], cForest [84, 85] and pForest are applied to the two data sets. Each forest used 200 trees. For the pForest test, set $\delta = 20\%$ and $R = 200$. Because Chi-square works only for categorical attributes, the continuous predictor variables were removed before Chi-square importance measure were applied.

6.4.1. Results from experiments without interactions

The feature importance scores of all data sets for the experiment without interactions are shown as box-plots in Figure 20. Figure 20(a) illustrates the expected pattern.

For basic RF in Figure 20(b), the importance measure prefers higher attributes variables for both informative and irrelevant variables. Also, it can't discriminate between X_5 and X_6. Furthermore, the continuous variable X_1 has the greatest importance score among the informative variables. However, for the irrelevant variables, the importance scores of the categorical attributes increase as the cardinality increases, and exceed the importance score of the continuous attribute when the cardinality equals 64.



(a) expected pattern (b) Random forest (c) cForest

(d) Chi-square (e) OOB forest (f) pForest

Figure 20. Feature importance from different methods for experiments without interactions. X axis represents variables, and Y axis represents importance score. Figure 20(a) illustrates the expected pattern, and the importance score in Figure 20(a) only represents a relative measure for comparing different variables.

For cForest in Figure 20(c), there is no bias among the irrelevant variables, which is consistent with the null case in [85]. cForest can also discriminate informative variables from irrelevant variables (though the differences between $Imp(X_5), Imp(X_6)$ and the U_k are not obvious), which is also consistent with the power case in [85]. However, for the informative variables, cForest prefers higher cardinality variables. Furthermore, it can not discriminate X_5 from X_6.

For Chi-square in Figure 20(d), higher-cardinality attributes are preferred for irrelevant variables. However, it is able to rank informative variables higher than irrelevant variables. Furthermore, it perform well for those informative variables, that is, there is no obvious multi-valued problems for informative variables.

For OOBForest in Figure 20(e), there is no bias in both informative and irrelevant variables. The expected orders among all predictor variables are well preserved. Therefore, OOBForest has good performance here.

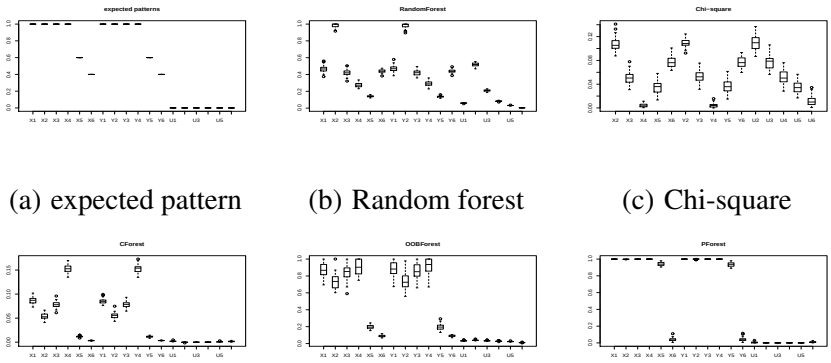
For pForest in Figure 20(f), there is no bias in both informative and irrelevant variables. The expected orders among all predictor variables are well preserved. Therefore, pForest is also has good performance here.

6.4.2. Results from experiments with interactions

The feature importance scores of all data sets for the experiment with interactions are shown as box-plots in Figure 21. Figure 21(a) illustrates the expected pattern.

For RF in Figure 21(b), the bias is even more severe than in the previous ex-

periment. The random forest cannot even discriminate irrelevant attributes from some informative variables. The importance of U_2 is only less than X_2 and Y_2. Therefore, the variable importance scores from random forest are extremely unreliable here.



(a) expected pattern (b) Random forest (c) Chi-square

(d) cForest (e) OOB forest (f) pForest

Figure 21. Feature importance from different methods for experiments with interactions. X axis represents variables, and Y axis represents importance score. Figure 21(a) illustrates the expected pattern, and the importance score in Figure 21(a) represents a relative measure for comparing different variables.

For Chi-square in Figure 21(c), it cannot even distinguish between the informative variables and the irrelevant variables. This is expected because Chi-square does not consider the interactions. This observation can be extended to other methods such as information gain, which only consider dependency between a single predictor variable and the target variable. It should noted that although random forest uses information criteria for each node, the summary of all nodes in the forest considers the interactions between the predictor variables.

For cForest in Figure 21(d), there is no obvious bias among the irrelevant variables and cForest can also discriminate the informative variables from the irrelevant variables (although the importance difference between X_6 and $U_i s$ is not obvious). However, there is multi-valued problem in the informative variables. In contrast to the previous experiment, cForest now prefers lower cardinality attributes.

For OOBForest in Figure 21(e), there is no bias among the irrelevant variables. The four groups can be discriminated. There are some minor importance differences among the most informative variables.

For pForest in Figure 21(f), there is no bias in both informative and irrelevant variables. The expected orders among all predictor variables are well preserved.

6.4.3. Selection of δ

Selecting a suitable δ is important for pForest to produce importance values that match the true relative relevance of variables. If δ is too small, then the pForest importance values for all variables tend to be small; and conversely, if δ is too large, then all variables relevant to the class could have similarly large importance values. An example is simulated to show the effect of δ selection. First, generate a two-level categorical variable V, with equal number of cases at each level. The target variable T is then derived from V with 5% rows randomly permuted, and the predictor variables V_1, V_2 are derived from V with 20% and 40% rows randomly permuted. Therefore, both V_1 and V_2 are relevant to T, and $Imp(V_1) > Imp(V_2)$. Here 100 cases are simulated, and

pForest is used to measure the importance of V_1 and V_2 with varying δ. The experiment is simulated 20 times. Figure 22 shows the average and standard errors of the pForest importance over the 20 replicates at each δ value. It can be seen that the importance values of V_1 and V_2 are similar when δ is too small or too large, and thus the importance values do not match the true relationship: $Imp(V_1) > Imp(V_2)$. Here 0.05 or 0.1 are good choices for δ.

As seen from the example, the ability of pForest to distinguish small difference in variable importance can depend on δ. A solution for selecting δ for a data set might be testing different δ values, and selecting a δ value maximizing the variability of variable importance values. This might better identify differences between the importance values. However, this enhancement is not studied further here.

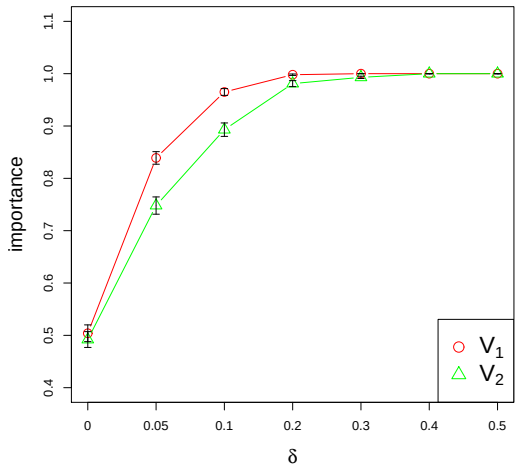


Figure 22. Selection of δ.

6.4.4. Discussion

From the experiments, it can be seen that RF is not reliable for feature selection when predictor variables have different cardinality, it always prefers high cardinality variables. cForest performs well for those irrelevant variables with different cardinality. However, it is not reliable enough for measuring the importance of informative variables with different cardinalities. Chi-square performs well regarding those unequally informative variables when interactions are not present. However, it prefers higher-cardinality ones for irrelevant variables. More importantly, Chi-square is not reliable when interactions are present. The results of pForest are much better than random forest. OOBForest performs also well in both experiments.

6.5. Conclusions

The bias of attribute importance measures is an important problem. In particular, the common use of RF for attribute importance is shown to be a concern. Two methods are proposed to solve the bias problem. One is based on out-of-bag samples [58], while the other method uses the new concept of a partial permutation test to refine the attribute importance scores. The second method is studied with an RF, but can be easily adapted to other feature scoring algorithms. The bias problem is studied in a simulation framework that integrates different cardinalities, and where non-equally informative attributes (with or without interactions) and irrelevant attributes co-exist. The proposed methods are compared directly to two existing solutions for multi-value bias: Chi-square and a

conditional inference framework. The experiments show that the existing methods are not always reliable for multi-valued predictors, while the proposed methods compare favorably.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1. Conclusions

This dissertation transforms a set of system complexity reduction problems to feature selection problems, which makes the original problems much easier to solve. Three systems are considered: classification based on association rules, network structure learning, and time series classification. Furthermore, two variable importance measures are proposed to reduce the feature selection bias in tree models.

Associative classification rule pruning and summarizing have been active research topics. While previous pruning methods focused on pruning rules to some degree, this research proposes to prune rule conditions into a minimum subset without loss of information regarding predicting the class, which suggests a new direction for rule pruning. In addition, although previous associative classifiers were derived from simple association rules, the large number of rules used in the final model results in limited interpretability. The rule-based classifiers discussed here consist of a significantly smaller number of rules without loss of accuracy, comparing to a well-known associative classifier in the experiments conducted here.

Network structure learning is important for both discrete and continuous networks, and relaxed Gaussian assumptions are important for continuous networks. By using minimum-relevancy-maximum redundancy feature selection methods, the network local structure can be learned without any restrictions on the distribution of the variables. Since tree ensembles are able to deal with complex data structures, a tree

ensemble feature selection method is proposed to learn local structures for discrete or continuous networks. The experiments conducted in this research illustrate the tree ensemble method is superior to a Bayesian structure learning algorithm, and other feature selection methods.

A time series forest (TSF) for time series classification, and feature extraction is proposed. TSF has computational complexity linear in the length of a time series, but is as accurate as widely used alternatives such as one-nearest-neighbor with the dynamic time warping metric. Furthermore, a scalable size of features can be extracted from TSF, and can be further reduced and summarized for better interpretability. The effectiveness, and efficiency are demonstrated by testing on a full set of benchmark data sets from UCR time series database [100]. The benchmark data sets are univariate, but the method can be also used for multivariate time series classification.

The bias of attribute importance measures is an important problem. Two methods to solve the bias problem are proposed. One is based on out-of-bag samples, while the other method uses the new concept of a partial permutation test to refine the attribute importance scores. The second method is studied with a RF, but can be easily adapted to other feature scoring algorithms. The proposed methods are compared to two existing solutions for multi-value bias: Chi-squared and a conditional inference framework. The experiments show that the existing methods are not always reliable for multi-valued predictors, while the proposed methods have advantages.

7.2. Future work

Feature selection is used for rule condition subset selection after transforming the rule condition set to a new data set. Since the predictor variables of the new data set are binary, it would be worthwhile to study alternative feature selection methods more efficient for binary data sets.

A feature selection method based on tree ensembles (ACE) is proposed to learn MBs of Bayesian networks. The experiments conducted here show the effectiveness of the method, still theoretical analysis is desired to understand why ACE works. In addition, one objective of learning MBs of a Bayesian network is to help identify a directed acyclic graph. Therefore, it is valuable to investigate if an approach similar to ACE could be developed to learn the directions of arcs in a network.

For time series classification, one disadvantage of TSF proposed here is that it is targeting time series with equal lengths. Though the time series can be aligned to the same lengths before applying TSF, still improvement on handling time series of different lengths would be desirable. As DTW has been considered one of the best solutions for handling time series with different lengths, it would be worthwhile to integrate DTW into feature-based methods without loss of much interpretability.

Potentially selecting the best interval feature at each node in TSF can be formulated as an optimization problem: the solution space consists of all the interval features, and the objective function is the evaluation criteria in TSF. Considering that interval features extracted from time series are closely related, it is valuable to find an efficient

method to solve the optimization problem.

Both OOBForest and pForst considered here can be used to measure attribute importance. While OOBForest is also a classifier less biased than random forest, pForst considered so far can be only used for measuring attribute importance. Therefore, it would be worthwhile to find a way to build a less-biased tree model by using the pForest concept. Furthermore, future work on reducing the computational complexity of pForest would be desired.

REFERENCES

[1] B. Liu, W. Hsu, and Y. Ma, "Integrating classification and association rule mining," in *KDD-98*, Aug 1998.

[2] B. Liu, Y. Ma, and C. Wong, "Classification using association rules: weaknesses and enhancements," *Data mining for scientific applications*, 2001.

[3] W. Li, J. Han, and J. Pei, "Cmar: accurate and efficient classification based on multiple class-association rules," in *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, Nov-Dec 2001, pp. 369–376.

[4] X. Yin and J. Han., "Cpar : Classification based on predictive association rules," in *Proceedings 3rd SIAM Int. Conf. on Data Mining SDM03*, 2003.

[5] I. Tsamardinos, C. Aliferis, and A. Statnikov, "Algorithms for large scale markov blanket discovery," in *Proceedings of the 16th International FLAIRS Conference*, 2003.

[6] L. Frey, D. Fisher, I. Tsamardinos, C. Aliferis, and A. Statnikov, "Identifying markov blankets with decision tree induction," *Third IEEE International Conference on Data Mining*, pp. 59 – 66, Nov 2003.

[7] J.-P. Pellet and A. Elisseeff, "Using Markov blankets for causal structure learning," *Journal of Machine Learning Research*, vol. 9, pp. 1295–1342, 2008.

[8] R. Tillman, A. Gretton, and P. Spirtes, "Nonlinear directed acyclic structure learning with weakly additive noise models," *Advances in Neural Information Processing Systems (NIPS)*, vol. 22, p. 18471855, 2009.

[9] H. Deng, S. Davila, G. C. Runger, and E. Tuv, "Learning markov blankets for continuous or discrete networks via feature selection," in *Proceedings of ECML-SUEMA 2010*, O. Okun, M. Re, and G. Valentini, Eds., Barcelona, Spain, 2010, pp. 97–108.

[10] J. Shieh and E. Keogh, "Polishing the right apple: Anytime classification also benefits data streams with constant arrival times," in *ICDM 2010*, 2010.

[11] I. Batal, L. Sacchi, R. Bellazzi, and M. Hauskrecht, "Multivariate Time Series Classification with Temporal Abstractions," *International journal of artificial intelligence tools: architectures, languages, algorithms*, vol. 22, p. 344, 2009.

[12] J. Caiado, N. Crato, and D. Peña, "A periodogram-based metric for time series classification," *Computational Statistics & Data Analysis*, vol. 50, no. 10, pp. 2668–2684, 2006.

[13] H. Sakoe, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, pp. 43–49, 1978.

[14] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. Ratanamahatana, "Fast time series classification using numerosity reduction," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 1033–1040.

[15] C. Ratanamahatana and E. Keogh, "Three myths about dynamic time warping data mining," in *Proceedings of SIAM International Conference on Data Mining (SDM05)*. Citeseer, 2005.

[16] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer, "Permutation importance: a corrected feature importance measure," *Bioinformatics*, vol. 26, no. 10, pp. 1340–1347, May 2010.

[17] H. Deng, G. C. Runger, and E. Tuv, "System monitoring with real-time contrasts," *Journal of Quality Technology*, 2010, accepted.

[18] A. P. White and W. Z. Liu, "Technical note: Bias in information-based measures in decision tree induction," *Machine Learning*, vol. 15, no. 3, pp. 321–329, June 1994.

[19] H. Deng, G. C. Runger, and E. Tuv, "Bias of importance measures for multi-valued attributes and solutions," in *Proceedings of ICANN 2011*.

[20] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, vol. 1215, 1994, pp. 487–499.

[21] C. Aggarwal and P. Yu, "Online generation of association rules," in *Proceedings of the international conference on data engineering*, 1998, pp. 402–411.

[22] R. Hilderman, C. Carter, H. Hamilton, and N. Cercone, "Mining market basket data using share measures and characterized itemsets," *Research and Development in Knowledge Discovery and Data Mining*, pp. 159–173, 1998.

[23] B. Liu, W. Hsu, and Y. Ma, "Mining association rules with multiple minimum supports," in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1999, pp. 337–341.

[24] J. Han and N. Cercone, "AViz: A visualization system for discovering numeric association rules," *Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications*, pp. 269–280, 2000.

[25] J. Vaidya and C. Clifton, "Privacy preserving association rule mining in vertically partitioned data," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2002, pp. 639–644.

[26] W. Wang, J. Yang, and P. Yu, "WAR: weighted association rules for item intensities," *Knowledge and Information Systems*, vol. 6, no. 2, pp. 203–229, 2004.

[27] X. Wu, C. Zhang, and S. Zhang, "Efficient mining of both positive and negative association rules," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 3, p. 405, 2004.

[28] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," *Data mining and knowledge discovery*, vol. 8, no. 1, pp. 53–87, 2004.

[29] B. Goethals, J. Muhonen, and H. Toivonen, "Mining non-derivable association rules," in *Proc. SIAM Int. Conf. on Data Mining*, 2005.

[30] W. Au and K. Chan, "Mining changes in association rules: a fuzzy approach," *Fuzzy sets and systems*, vol. 149, no. 1, pp. 87–104, 2005.

[31] A. Amir, Y. Aumann, R. Feldman, and M. Fresko, "Maximal association rules: a tool for mining associations in text," *Journal of Intelligent Information Systems*, vol. 25, no. 3, pp. 333–345, 2005.

[32] T. Gharib, H. Nassar, M. Taha, and A. Abraham, "An efficient algorithm for incremental mining of temporal association rules," *Data & Knowledge Engineering*, 2010.

[33] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference and prediction*, 2nd ed. Springer, 2009. [Online]. Available: http://www-stat.stanford.edu/~tibs/ElemStatLearn/

[34] F. Thabtah, "Pruning techniques in associative classification: Survey and comparison," *Journal of Digital Information Management*, vol. 4, no. 3, p. 197, 2006.

[35] ——, "A review of associative classification mining," *The Knowledge Engineering Review*, vol. 22, no. 01, pp. 37–65, 2007.

[36] M. Antonie, O. Zaïane, and A. Coman, "Associative classifiers for medical images," *Mining Multimedia and Complex Data*, pp. 68–83, 2003.

[37] F. Thabtah, P. Cowling, and Y. Peng, "MMAC: A New Multi-class, Multi-label Associative Classification Approach," in *Proceedings of the 4th IEEE International Conference on Data Mining, ICDM '04*, 2004, pp. 217–224.

[38] O. Zaïane and M. Antonie, "On pruning and tuning rules for associative classifiers," in *Knowledge-Based Intelligent Information and Engineering Systems*. Springer, 2005, pp. 966–973.

[39] W. Bannister, "Associative and sequential classification with adaptive constrained regression methods," Ph.D. dissertation, Arizona State University, 2007.

[40] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[41] H. Cheng, X. Yan, J. Han, and C. Hsu, "Discriminative frequent pattern analysis for effective classification," in *2007 IEEE 23rd International Conference on Data Engineering*. IEEE, 2007, pp. 716–725.

[42] N. Megiddo and R. Srikant, "Discovering predictive association rules," in *In Proc. of the 4th Int'l Conference on Knowledge Discovery in Databases and Data Mining*, 1998, pp. 274–278.

[43] J. R. Quinlan and R. M. Cameron-jones, "Foil: A midterm report," in *In Proceedings of the European Conference on Machine Learning*. Springer-Verlag, 1993, pp. 3–20.

[44] K. Wang, S. Zhou, and Y. He, "Growing decision trees on support-less association rules," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2000, pp. 265–269.

[45] J. Wang and G. Karypis, "HARMONY: Efficiently mining the best rules for classification," in *SIAM International Conference on Data Mining*, 2005, pp. 205–215.

[46] G. Chen, H. Liu, L. Yu, Q. Wei, and X. Zhang, "A new approach to classification based on association rule mining," *Decision Support Systems*, vol. 42, no. 2, pp. 674–689, 2006.

[47] F. Thabtah, P. Cowling, and Y. Peng, "MCAR: multi-class classification based on association rule," in *Computer Systems and Applications, 2005. The 3rd ACS/IEEE International Conference on*. IEEE, 2005, p. 33.

[48] Z. Tang and Q. Liao, "A new class based associative classification algorithm," *IAENG International Journal of Applied Mathematics*, vol. 36, no. 2, 2007.

[49] A. Veloso, W. Meira, and M. Zaki, "Lazy associative classification," in *Sixth International Conference on Data Mining, 2006. ICDM'06*, 2006, pp. 645–654.

[50] F. Thabtah, P. Cowling, and Y. Peng, "Multiple labels associative classification," *Knowledge and Information Systems*, vol. 9, no. 1, pp. 109–129, 2006.

[51] A. Veloso, W. Meira, M. Gonçalves, and M. Zaki, "Multi-label lazy associative classification," *Knowledge Discovery in Databases: PKDD 2007*, pp. 605–612, 2007.

[52] Y. Chen and L. Hung, "Using decision trees to summarize associative classification rules," *Expert Systems with Applications*, vol. 36, no. 2, pp. 2338–2351, 2009.

[53] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine learning*, vol. 29, no. 2, pp. 131–163, 1997.

[54] D. Koller and M. Sahami, "Toward optimal feature selection," in *Proceedings of ICML-96: 13th International Conference on Machine Learning*, 1996, pp. 284–292.

[55] D. Margaritis and S. Thrun, "Bayesian network induction via local neighborhoods," in *In Advances in Neural Information Processing Systems 12(NIPS)*, 1999.

[56] J. N. P. Pudil and J. Kittler, "Floating search methods in feature selection," *Pattern Recognition Letters*, vol. 15, no. 11, pp. 1119–1125, 1994.

[57] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine Learning*, vol. 46, no. 1-3, pp. 389–422, 2002.

[58] E. Tuv, A. Borisov, G. Runger, and K. Torkkola, "Feature selection with ensembles, artificial variables, and redundancy elimination," *Journal of Machine Learning Research*, vol. 10, pp. 1341–1366, 2009.

[59] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, Berlin, 2009.

[60] J. Lin, D. Etter, and D. DeBarr, "Exact and approximate reverse nearest neighbor search for multimedia data," in *Proc. SIAM International Conference on Data Mining*, 2008, pp. 656–667.

[61] K. Ueno and R. Orihara, "Time Series Classification by Peak-based Feature Extraction with DESSIN Representation," in *ACM SIGKDD '07-Workshop and Challenge on Time Series Classification*, 2007.

[62] D.-A. García-López and H.-G. Acosta-Mesa, "Discretization of time series dataset with a genetic search," in *MICAI*, 2009, pp. 201–212.

[63] Y. Jeong, M. Jeong, and O. Omitaomu, "Weighted dynamic time warping for time series classification," *Pattern Recognition*, 2010.

[64] E. Keogh and S. Kasetty, "On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration," *Data Mining and Knowledge Discovery*, vol. 7, no. 4, pp. 349–371, October 2003. [Online]. Available: http://dx.doi.org/10.1023/A:1024988512476

[65] C. Ratanamahatana and E. Keogh, "Making time-series classification more accurate using learned constraints," in *Proceedings of SIAM International Conference on Data Mining*. Lake Buena Vista, Florida, 2004, pp. 11–22.

[66] K. Ueno, X. Xi, E. Keogh, and D. Lee, "Anytime classification using the nearest neighbor algorithm with applications to stream mining," in *Data Mining, 2006. ICDM'06. Sixth International Conference on*. IEEE, 2007, pp. 623–632.

[67] Z. Xing, J. Pei, and P. Yu, "Early prediction on time series: a nearest neighbor approach," in *Proceedings of the 21st international jont conference on Artifical intelligence*. Morgan Kaufmann Publishers Inc., 2009, pp. 1297–1302.

[68] Y. Yamada, H. Yokoi, and K. Takabayashi, "Decision-tree induction from time-series data based on standard-example split test," in *In Proceedings of the 20th International Conference on Machine Learning (ICML03*. Morgan Kaufmann, 2003, pp. 840–847.

[69] D. Eads, K. Glocer, S. Perkins, and J. Theiler, "Grammar-guided feature extraction for time series classification," in *NIPS 05*, 2005.

[70] M. Hearst, S. Dumais, E. Osman, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and Their Applications*, vol. 13, no. 4, pp. 18–28, 1998.

[71] A. Nanopoulos, R. Alcock, and Y. Manolopoulos, "Feature-based classification of time-series data," *International Journal of Computer Research*, vol. 10, pp. 49–61, 2001.

[72] J. Rodríguez, C. Alonso, and J. Maestro, "Support vector machines of interval-based features for time series classification," *Knowledge-Based Systems*, vol. 18, no. 4-5, pp. 171–178, 2005.

[73] J. Rodríguez, C. Alonso, and H. Boström, "Boosting interval based literals," *Intelligent Data Analysis*, vol. 5, no. 3, pp. 245–262, 2001.

[74] J. Rodríguez and C. Alonso, "Interval and dynamic time warping-based decision trees," in *Proceedings of the 2004 ACM symposium on Applied computing*. ACM, 2004, p. 552.

[75] V. Eruhimov, V. Martyanov, and E. Tuv, "Feature Class Selection for Time Series Classification," in *ACM SIGKDD '07-Workshop and Challenge on Time Series Classification*, 2007.

[76] V. Eruhimov, V. Martyanov, P. Raulefs, and E. Tuv, "Combining unsupervised and supervised approaches to feature selection for multivariate signal compression," *Intelligent Data Engineering and Automated Learning–IDEAL 2006*, pp. 480–487, 2006.

[77] A. Borisov, V. Eruhimov, and E. Tuv, "Tree-based ensembles with dynamic soft feature selection," in *Feature Extraction Foundations and Applications: Studies in Fuzziness and Soft Computing*, M. N. I. Guyon, S. Gunn and e. L. Zadeh, Eds.

[78] D. Minnen, P. Zang, C. Isbell, and T. Starner, "Boosting Diverse Learners for Domain Agnostic Time Series Classification," in *ACM SIGKDD '07-Workshop and Challenge on Time Series Classification*, 2007.

[79] J. Rodríguez and L. I. Kuncheva, "Time Series Classification: Decision Forests and SVM on Interval and DTW Features," in *ACM SIGKDD '07-Workshop and Challenge on Time Series Classification*, 2007.

[80] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Wadsworth, Belmont, MA, 1984.

[81] K. Igor, "On biases in estimating multi-valued attributes," in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, Montral, Canada*, 1995, pp. 1034–1040.

[82] K. Kira and L. A. Rendell, "A practical approach to feature selection," in *Proceedings of the ninth international workshop on Machine learning, Aberdeen, Scotland, United Kingdom*, 1992, pp. 249 – 256.

[83] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[84] T. Hothorn, K. Hornik, and Z. Achim, "Unbiased recursive partitioning: A conditional inference framework," *Journal of Computational and Graphical Statistics*, vol. 15, pp. 651–674, JAN 2006.

[85] C. Strobl, A.-L. Boulesteix, A. Zeileis, and T. Hothorn, "Bias in random forest variable importance measures: Illustrations, sources and a solution," *BMC Bioinformatics*, vol. 8, no. 25, JAN 2007.

[86] C. Blake and C. Merz, "UCI repository of machine learning databases," 1998.

[87] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling, "Numerical recipes in c," 1988.

[88] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[89] M. A. Hall, "Correlation-based feature selection for discrete and numeric class machine learning," in *Proceedings of the 17th International Conference on Machine Learning*, 2000, pp. 359–366.

[90] L. Yu and H. Liu, "Efficient feature selection via analysis of relevance and redundancy," *Journal of Machine Learning Research*, vol. 5, pp. 1205–1224, 2004.

[91] R. Ihaka and R. Gentleman, "R: A language for data analysis and graphics," *Journal of computational and graphical statistics*, vol. 5, no. 3, pp. 299–314, 1996.

[92] K. Hornik, C. Buchta, and A. Zeileis, "Open-source machine learning: R meets Weka," *Computational Statistics*, vol. 24, no. 2, pp. 225–232, 2009.

[93] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. San Francisco: Morgan Kaufmann, 2005.

[94] U. M. Fayyad and K. B. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning," in *Proceedings of the International Joint Conference on Uncertainty in AI*. Springer-Verlag, 1993, pp. 1022–1027.

[95] J. Quinlan, *C4. 5: programs for machine learning*. Morgan Kaufmann, 1993.

[96] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[97] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[98] S. A. Tsamardinos I, Aliferis CF, "Time and sample efficient discovery of markov blankets and direct causal relations," in *Proceedings of the Ninth International Conference on Knowledge Discovery and Data Mining (KDD) 2003*, 2003, pp. 673–678.

[99] M. Scutari, "Learning bayesian networks with the bnlearn R package," *Journal of Statistical Software*, vol. 35, no. 3, pp. 1–22, 2010. [Online]. Available: http://www.jstatsoft.org/v35/i03/

[100] E. Keogh, X. Xi, L. Wei, and C. Ratanamahatana, "The ucr time series classification/clustering. homepage: www.cs.ucr.edu/~eamonn/time_series_data/," 2006. [Online]. Available: www.cs.ucr.edu/~eamonn/time_series_data/

[101] L. Yu and H. Liu, "Feature selection for high-dimensional data: A fast correlation-based filter solution," *Proceedings of the Twentieth International Conference on Machine Learning*, pp. 856–863, 2003.

[102] R. Genuer, J. Poggi, and C. Tuleau-Malot, "Variable selection using random forests," *Pattern Recognition Letters*, vol. 31, no. 14, pp. 2225–2236, 2010.

[103] P. Radivojac, Z. Obradovic, A. K. Dunker, and S. Vucetic, "Feature selection filters based on the permutation test," in *Machine Learning: ECML 2004, 15th European Conference on Machine Learning*, 2004.