SMT: Sparse Multivariate Tree[*]

Houtao Deng[†] Mustafa Gokce Baydogan [‡] George Runger [§]

Abstract

A multivariate decision tree attempts to improve upon the single variable split in a traditional tree. With the increase in data sets with many features and a small number of labeled instances in a variety of domains (e.g., bioinformatics, text mining, etc.), a traditional tree-based approach with a greedy variable selection at a node may omit important information. Therefore, the recursive partitioning idea of a simple decision tree combined with the intrinsic feature selection of L_1 regularized logistic regression at each node is a natural choice for a multivariate tree model that is simple, but broadly applicable. This natural solution leads to the sparse multivariate tree (SMT) considered here. SMT can naturally handle non-time-series data and is extended to handle time-series classification problems with the power of extracting interpretable temporal patterns (e.g., means, slopes, deviations). Binary L_1 regularized logistic regression models are used here for binary classification problems. However, SMT may be extended to solve multi-class problems with multinomial logistic regression models. The accuracy and computational efficiency of SMT is compared to a large number of competitors on time series and non-time-series data.

Keywords: time series classification; decision tree; Lasso; fused Lasso; feature extraction

1 Introduction

Decision tree classifiers are comprehensible models with satisfactory accuracy and are successfully used in many applications. Univariate trees such as CART [1] and C4.5 [2] split data based on only one variable at each node, and, thus, are limited to splits that are orthogonal to the variable's axis [3]. This limitation reduces the ability to express concepts succinctly [3]. Multivariate decision trees [3, 4] use a function of potentially more than one variable at a tree node and were proposed to overcome this limitation.

Multivariate trees generally are more concise than univariate trees regarding the tree depth. However, a function of many variables, potentially all the predictor variables, is formed at a tree node. When the number of variables is large, the combination can be difficult to interpret. To this end, procedures such as sequential backward elimination, sequential forward selection [3] and Tabu search [5] have been used to select a small subset of features in a node. These procedures use greedy algorithms and may require expensive computations (e.g., sequential backward elimination needs to evaluate $O(p)$ models, where p is the number of predictor variables). Therefore, it is desirable to develop an efficient, but less greedy method to produce a function of a small number of variables.

Time series classification is another challenging problem addressed here. 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. Feature-based classifiers build models on temporal features that capture local temporal characteristics. To create an interpretable classifier, domain knowledge can be integrated into feature extraction [6]. When domain knowledge is not available, general features may be preferred. [7, 8, 9] first extracted temporal features such as the mean of the variables over a time interval, and then trained classifiers such as neural networks and support vector machines on the features for classification.

Also, tree-ensemble methods that integrate feature extraction and classification have been proposed. For example, [10] built boosted binary stumps on temporal features from intervals of time series. There are $p \times (p-1)$ possible time intervals on which features can be extracted (where p is the time series length) which results in a temporal feature space of $O(p^2)$. Also, [10] reduced the temporal feature space to $O(p \log p)$ by considering intervals with lengths that are powers of two (i.e. $2, 4, 8, \ldots, \lfloor \log p \rfloor$). A time series forest (TSF) [11] used a random sampling strategy and further reduced the feature space to $O(p)$ without loss of accuracy. Though these ensemble methods use only simple features and are useful for feature extraction, the classifiers themselves

[*]This research was partially supported by ONR grant N00014-09-1-0656.

[†]Intuit, Mountain View, CA 94043, USA. Email: hdeng3@asu.edu

[‡]Department of Industrial Engineering, Boğaziçi University, 34342, İstanbul, Turkey. Email: mbaydoga@asu.edu

[§]Corresponding author. School of Computing, Informatics & Decision Systems Engineering, Arizona State University, Tempe, AZ 85287, USA. Email: george.runger@asu.edu

are difficult to interpret. The fused Lasso [12] which penalizes the L_1-norm of both the coefficients and their successive differences is a simple model that can be used for time series classification. It is designed for problems in which features are ordered in some meaningful way, e.g., time series. However it is limited to linear models and, only captures mean patterns in its basic form (as discussed in Section 4.2).

The sparse multivariate tree (SMT) framework proposed here aims to tackle the challenging problems mentioned above for two-class classification problems. SMT uses an L_1 regularized logistic regression (LR) model at each node and, therefore, forms a function of only a small number of variables. By using two types of L_1 regularized LR models, i.e., Lasso LR and fused Lasso LR, SMT can be used for classifying non-time-series and time-series data sets. For time series, we first extend the fused Lasso to capture slope and deviation patterns (in addition to mean patterns) that can be effective for classification. Then the fused Lasso LR models are integrated by SMT. Such a SMT classifier is interpretable and also useful for temporal feature extraction. SMT extends L_1 regularized LR to learn functions more complex than a linear logistic function. Although binary L_1 regularized LRs are considered in this study, SMT can be extended to solve multi-class problems with a multinomial logistic model.

The remainder of this paper is organized as follows. Section 2 presents related work. Section 3 introduces the base components for SMT. Section 4 describes the SMT models. Section 5 demonstrates the effectiveness and efficiency of SMT by extensive experiments. Conclusions are drawn in Section 6.

2 Related work

2.1 Multivariate tree classifiers

Univariate trees such as CART [1], and C4.5 [2] split on only one variable at each node. For multivariate trees, various methods have been used to combine multiple variables at a tree node: randomized hill-climbing algorithms (OC1 [13]), evolutionary algorithms [14], linear discriminant analysis (QUEST [15], Ltree [4], SURPASS [16], LDTS [5], LDT [17] and CRUISE [18]). Also, in an NBTree [19] a pre-pruning strategy based on a naive Bayes model was used at each node to decide whether to split further or to form a leaf node. A functional tree (FT) [20] built multivariate decision nodes when growing the tree, and formed functional leaves when pruning the tree.

Logistic regression has been used in tree models for supervised learning [21, 22]. LOTUS [21] applied logistic regression in tree nodes for two-class problems. LOTUS builds a tree in a top-down fashion and prunes the tree similar to the pruning algorithm used in CART [1]. A logistic model tree (LMT) [22] builds logistic regression functions at leaf nodes of a tree and prunes the tree similar to the CART pruning algorithm.

Generally multivariate trees have smaller depth than univariate trees. However, the function of many predictor variables at each node has potential to overfit. Methods such as Tabu search [5], sequential backward elimination and sequential forward selection [3] were used to reduce the number of variables. SMT proposed here does not require a separate variable selection procedure. It directly fits a function of a small subset of variables (sparse) at each node.

There are three main kinds of sparsity for prediction problems:

1. Only a small number of variables, say q, are relevant among the p variables.

2. All of the p variables are important, but we can partition the space so that in any local region, only a small number of variables are relevant.

3. All of the p variables are important, but we can find a small number of linear combinations of those variables that explain most of the variation in the response.

SMT addresses all of the items listed above. Recursive partitioning enabled by trees allows a focus to local regions where a small number of variables are relevant (item 2). Furthermore, L_1 regularized LR is used to find only a small number of variables that are relevant (item 1). Lastly, regression at each node of the tree only uses the small number of linear combinations of the variables that are important for prediction.

2.2 Time series classification

Time series classification and feature extraction have been active research topics in recent years. Instance-based classifiers such as one-nearest-neighbor (1NN) classifiers with a Euclidean distance metric (NNEuclidean) or a dynamic time warping metric (NNDTW) have been widely, and successfully used [23]. Although Euclidean distance is time and space efficient, it is often weak in terms of prediction accuracy [23]. DTW [24] allows a measure of the similarity independent of certain non-linear variations in the time dimension, and is considered a strong solution for time series problems [25]. Instance-based classifiers such as NNDTW can be difficult to interpret. The time series shapelets approach was proposed to address this issue [26, 27, 28]. The objective of time series shapelets is to find time series subsequences that are useful for separating different classes.

Subsequences unique to only a particular class can provide insights about that class.

Feature-based classifiers [6, 7, 8, 9, 10, 29, 30] build models on temporal features that can describe local temporal characteristics. However, the temporal feature space can be large [10, 31]. To reduce the computational complexity, several methods have been proposed to reduce the feature space. For example, [8, 10] extracted features only from intervals with length of powers of 2, and the feature space was reduced from p^2 to $p \log p$, where p is the time series length. These methods, however, can discard some potentially useful features. Furthermore, previous research focused on classification accuracy, and used complex models for classification (e.g., SVM [29], boosted trees [10]), while a simple and interpretable model with satisfactory accuracy is often desirable.

The interpretability of a time series classifier is highly important in various domains including medicine [32]. First, it is important to verify results in interdisciplinary teams of researchers. It is not easy to determine why an instance is assigned to a certain class when NN classification is used. However, a doctor would like to understand why such an assignment is made. Second, in applications such as fault diagnosis in manufacturing, it is useful to determine when and where a problem occurs [33]. In these applications, observations (such as the temperature after each processing stage) are collected during the stages of a manufacturing process. An interpretable classifier, trained to identify the normal and faulty products, can provide insights about where (which machine), when (at what stage) and in what conditions (the level of temperature) the problems occur.

3 Base components of SMT

3.1 L_1 regularized logistic regression

Denote a set of training instances as $(x_i; y_i)$, $i = 1, ..., N$, where $y_i \in \{-1, +1\}$ are binary outcomes and $x_i = (x_{i1}, ..., x_{ip})^T$, where x_{ij} are realizations of predictor variables $X_j, j = 1, ..., p$. For time series data, $X_1, X_2, ..., X_p$ are ordered by the time index, and p is the length of the time series.

In a logistic regression (LR) model, the probability of y_i being class 1 given x_i is

$$p(y_i = 1|x_i, \beta, \beta_0) = \frac{\exp(\beta_0 + \beta^T x_i)}{1 + \exp(\beta_0 + \beta^T x_i)} \tag{1}$$

where $\beta = \{\beta_1, \beta_2, ..., \beta_p\}^T$ are the coefficients of the LR model, and β_0 is the intercept.

To calculate the values of β and β_0, maximum likelihood estimation is commonly used. However, we consider penalty terms and the objective function

$$\min_{\beta, \beta_0} \{ \sum_{i=1}^{N} \log(1 + \exp(-y_i(\beta_0 + \beta^T x_i))) + \lambda_1 ||\beta||_1 + \lambda_2 \sum_{j=1}^{p-1} ||\beta_j - \beta_{j+1}||_1 + \lambda_3 ||\beta||_2 \} \tag{2}$$

where $|| \cdot ||_1$ and $|| \cdot ||_2$ denote the L_1 and L_2 norms, respectively. Denote $\hat{\beta}$ and $\hat{\beta}_0$ as the estimates for β and β_0 in equation (2). When $\lambda_1 = \lambda_2 = \lambda_3 = 0$, equation (2) is an ordinary LR problem. When N, the number of training instances, is not large enough compared to p, ordinary LR may lead to over-fitting [34]. To this end, regularization methods penalizing large β were proposed [12, 35]. When $\lambda_1 = \lambda_2 = 0$, $\lambda_3 > 0$, equation (2) becomes L_2 regularized logistic regression, which produces a shrunk $\hat{\beta}$. However, potentially all the entries of $\hat{\beta}$ can be nonzero, which may be difficult to interpret when p is large.

For L_1 regularized LR, (i.e. $\lambda_1 > 0$, $\lambda_3 = 0$), the aim is to penalize the number of nonzero entries. Here L_1 regularized LR with $\lambda_2 = 0$ is called Lasso LR [35], and L_1 regularized LR with $\lambda_2 > 0$ is called fused Lasso LR [12]. Fused Lasso LR differs from Lasso LR in the way it penalizes both the L_1 norm of the coefficients and their successive differences [12], i.e., $|\beta_j - \beta_{j+1}|$. The fused Lasso model was designed for problems with features that can be ordered in some meaningful way, e.g., time series. Efficient algorithms for solving regularized LR can be found in [34, 36]. We use a solver [37] for convex optimization problems such as equation (2).

Figure 1 illustrates $\hat{\beta}/10$ from ordinary LR, $\hat{\beta}$ from Lasso LR ($\lambda_1 = 0.3$) and fused Lasso LR ($\lambda_1 = \lambda_2 = 0.3$) for the ECG time series classification problem [38], which includes 100 training instances and 96 features. It can be seen from Figure 1 that most of the coefficients from ordinary LR are nonzero. Only a small number of coefficients have nonzero values for Lasso LR and fused Lasso LR. The coefficients of successive time points tend to be the same for fused Lasso LR.

3.2 Splitting and pruning functions

Splitting and pruning functions are two key components in decision tree models. A splitting function partitions the data in a node into child nodes. A pruning function prunes a decision tree to prevent over-fitting. We describe two splitting functions as well as a pruning function for SMT.

The information gain criterion [2] is widely used for splitting in decision trees. Denote the proportions of instances with the corresponding classes at a node as $\{\nu_1, \nu_2, ...\nu_C\}$ (where $C = 2$ for binary classification).

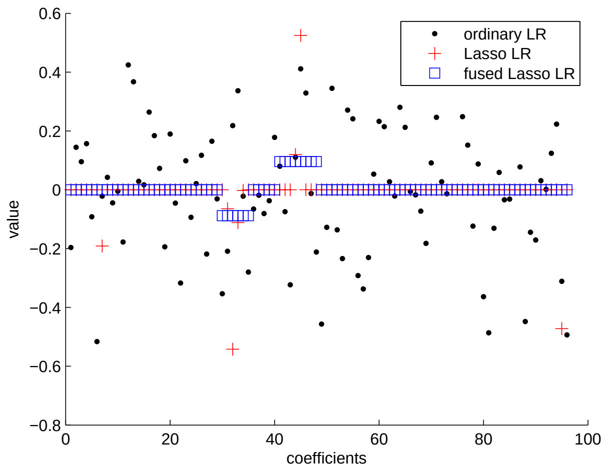


Figure 1: An illustration of differences between ordinary LR, Lasso LR, and fused Lasso LR. Coefficient values $\hat{\beta}/10$ from ordinary LR, $\hat{\beta}$ from Lasso LR ($\lambda_1 = 0.3$) and fused Lasso LR ($\lambda_1 = \lambda_2 = 0.3$) for the ECG time series classification problem are shown. Most entries of $\hat{\beta}$ from ordinary LR are nonzero. Both Lasso LR and fused Lasso LR produce a sparse $\hat{\beta}$, and the successive nonzero coefficients from fused Lasso LR tend to be equal.

The entropy at the node is defined as

$$Entropy = -\sum_{c=1}^{C} \nu_c \log \nu_c \tag{3}$$

The information gain $\triangle Entropy$ for a split is the difference between the weighted sum of the entropies at the child nodes and the entropy at the parent node, where the weight at a child node is the proportion of the instances assigned to that child. To find the splitting point of a (numerical) predictor variable V, the instances are ordered by V. Then the splitting point is the midpoint of the two consecutive values leading to the maximum $\triangle Entropy$.

An alternative splitting criterion is based on the value of a LR model that is fit at each node of the tree. That is, given estimates from problem (2), define the variable $V = \hat{\beta}_0 + \hat{\beta}^T x$. An instance x is assigned by LR to a class based on the sign of V, and a split can simply be based on $V > 0$. This split (suggested by an anonymous reviewer) is compared to the entropy alternative applied to V from LR in the experiments.

To prevent overfitting, post-pruning algorithms are commonly used in tree models. Post-pruning algorithms prune a fully-grown decision tree (typically) in a bottom-up fashion. The pruning algorithm used in C4.5 [2] is considered here. The pessimistic error rate estimate at a node is calculated as [39]

$$e = \frac{f + \frac{z^2}{2n} + z\sqrt{\frac{f}{n} - \frac{f^2}{n} + \frac{z^2}{4n^2}}}{1 + \frac{z^2}{n}} \tag{4}$$

where n is the number of instances at the node, f is the observed error rate, and z is the upper α percentile of a standard normal distribution. Here we set $\alpha = 0.25$, i.e., $z = 0.69$, the same as the default setting in Weka software [40]. The pessimistic error rate of the subtree of a node is defined as the weighted sum of the pessimistic error rates from all the leaf nodes in the subtree, where the weight is proportional to the number of instances at the leaf node. The subtree of a node is pruned if the pessimistic error rate of the node is smaller than its subtree. More details about the pruning algorithm were provided by[39].

3.3 Handling of missing values and nominal attributes

Missing values are approximated before running SMT. The missing values are simply replaced with the median (for numeric features) or mode (for categorical features) of the respective feature based on the training data. The same medians and modes are used to replace the missing values in the test data. Alternative ways to handle the missing values are further discussed by [41].

Furthermore, nominal features are transformed into binary features before running SMT. A nominal feature with k levels is transformed into $k-1$ binary indicator features and treated as numeric as in [22]. This increases the number of features. However, L_1 regularized logistic regression models can be solved efficiently even with a large number of features [37].

4 Sparse multivariate tree

4.1 Basic algorithm

The SMT algorithm integrates L_1 regularized LRs in a tree using a splitting and pruning function from the previous section. Denote X as a $N \times p$ matrix with the ijth entry x_{ij}, and $y = (y_1, ..., y_N)^T$. Algorithm 1 shows the steps to build an unpruned SMT. In Algorithm 1, Steps 1-4 are related to the stopping criteria when the node cannot be further divided. A node that cannot be split is designated as a leaf node with class label equal to the majority class of the instances in the node. Step 5 fits a L_1 regularized LR model. A model using Lasso LR is called a **SMT with Lasso LR**. A model using fused Lasso LR is called a **SMT with fused Lasso LR**. Step 6 forms a new variable based on the regularized LR model just built. Step 8 calculates the best splitting point from the new variable, either from entropy or simply from the LR threshold $V = 0$. Steps 9-12 split the instances to either the left or right child node. After building an unpruned SMT, the C4.5 pruning algorithm mentioned previously is used to prune the tree to a SMT model. Each leaf node is labeled with the majority class of the training instances in that node, and, thus, any testing instance assigned to the leaf node is assigned to the labeled class. We use cross-validation to select the regularization parameters. Here we set the same regularization parameters for all the tree nodes. Alternatively, one can select different regularization parameters at each node.

A SMT with Lasso LR can be used directly for non-time-series classification. Though a SMT with fused Lasso LR can be used for time-series data, additional procedures enhancing the model are introduced in the next section. It should be noted that only binary L_1 regularized LRs are considered here, potentially SMT can be extended with a multinomial logistic model for problems with multiple classes.

Algorithm 1 The basic SMT algorithm: $tree(D)$, where $D = [X, Y]$, where X is a $N \times p$ matrix with ijth entry x_{ij}, and $Y = (y_1, ..., y_N)^T$

```
 1: if can not further divide then
 2:     designate this node as a leaf node with class label equal to the majority class of the instances in the node
 3:     return
 4: end if
 5: obtain $\hat{\beta}$ and $\hat{\beta}_0$ by solving problem (2) for $D = [X, Y]$
 6: $V = X\hat{\beta} + \hat{\beta}_0$
 7: $D' = [V, Y]$
 8: obtain $V^*$, the splitting point of variable $V$, to partition $D'$
 9: $D_{left} \leftarrow$ instances $x_i$ with $\hat{\beta}^T x_i + \hat{\beta}_0 \leq V^*$
10: $D_{right} \leftarrow$ instances $x_i$ with $\hat{\beta}^T x_i + \hat{\beta}_0 > V^*$
11: $tree(D_{left})$
12: $tree(D_{right})$
```

4.2 Time series classification and feature learning

Now we adapt Algorithm 1 to an effective time series classifier and feature learner. First define new matrices X^a, X^b and X^c, where $X^a = X$, the columns of X^b are $X^b_j = X_{j+1} - X_j$, and similarly for X^c, $X^c_j = |X_{j+1} - X_j|$, $j = 1, 2, ..., p-1$. The algorithm for time series classification is shown in Algorithm 2. Note that Algorithm 2 differs from Algorithm 1 in the way that three matrices

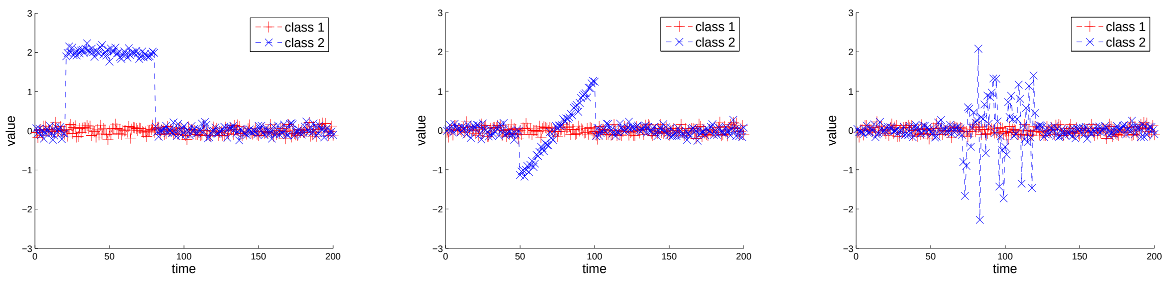
Algorithm 2 The SMT algorithm for time series: $\widetilde{tree}(D)$, where $D = [X^a, X^b, X^c, Y]$, where $X^a = X$, and $X^b_j = X_{j+1} - X_j$, and $X^c_i = |X_{j+1} - X_j|$, $j = 1, 2, ..., p-1$, where X is a $N \times p$ matrix with ijth entry x_{ij}, and $Y = (y_1, ..., y_N)^T$

```
 1: if can not further divide then
 2:     designate this node as a leaf node with class label equal to the majority class of the instances in the node
 3:     return
 4: end if
 5: for $\gamma$ in $\{a, b, c\}$ do
 6:     obtain $\hat{\beta}^\gamma$ and $\hat{\beta}^\gamma_0$ by solving equation (2) for $[X^\gamma, Y]$
 7:     $V^\gamma = X^\gamma \hat{\beta}^\gamma + \hat{\beta}^\gamma_0$
 8:     $D'^\gamma = [V^\gamma, Y]$
 9:     use the split $V^\gamma > 0$ with gain $\Delta Entropy^\gamma$ to partition $D'^\gamma$
10: end for
11: $\gamma^* = \arg\max_\gamma \Delta Entropy^\gamma$
12: $D_{left} \leftarrow$ instances $x_i$ with $(\hat{\beta}^{\gamma^*})^T x^{\gamma^*}_i + \hat{\beta}^{\gamma^*}_0 \leq 0$
13: $D_{right} \leftarrow$ instances $x_i$ with $(\hat{\beta}^{\gamma^*})^T x^{\gamma^*}_i + \hat{\beta}^{\gamma^*}_0 > 0$
14: $\widetilde{tree}(D_{left})$
15: $\widetilde{tree}(D_{right})$
```

X^a, X^b, X^c are used to calculate the split at a node. From the regularized LR model applied to each matrix, variables V^γ for $\gamma \in \{a, b, c\}$ are obtained at each node. Each of the three potential splits $V^\gamma > 0$ is scored with information gain, , or the split value is selected based on entropy, and the split with the best information gain is selected.

The motivation for the new matrices follows from the following simple properties. The fused Lasso LR penalizes the difference between the coefficients at successive time points, and, therefore, the coefficients from the fused Lasso LR tend to be same at successive time points. Denote $\hat{\beta}_j^{\ \gamma}$ as the jth entry of $\hat{\beta}^\gamma$ estimated by fused Lasso LR on matrix $[X^\gamma, Y]$ and denote x^γ_{ij} as the ijth entry of X^γ, where $\gamma \in \{a, b, c\}$. According to the definition of X^a, X^b and X^c, we have $x^a_{ij} = x_{ij}$, $x^b_{ij} = x_{i(j+1)} - x_{ij}$, $x^c_{ij} = |x_{i(j+1)} - x_{ij}|$, for $j = 1, 2, ..., p-1$. Then at a SMT node, the fused Lasso LR applied to the new matrices generates the following properties through equality of the coefficients.

If $\hat{\beta}^a_j = \hat{\beta}^a_{j+1} = ... = \hat{\beta}^a_k \neq 0$, then $\hat{\beta}^a_j x^a_{ij} + ... + \hat{\beta}^a_k x^a_{ik} = \hat{\beta}^a_j (x_{ij} + ... + x_{ik})$. Therefore, the mean of the values in the interval between time j and time k may be informative for classification. In such a case, the fused Lasso LR learns a **mean pattern**. An example is shown in Figure 2(a) where the mean of the values in the interval between time index 20 and 80 is informative to distinguish the two



(a) mean pattern (b) slope pattern (c) deviation pattern

Figure 2: Illustrations of interpretable patterns effective for classifying time series.

time series.

If $\hat{\beta}^b_j = \hat{\beta}^b_{j+1} = ... = \hat{\beta}^b_{k-1} \neq 0$, then $\hat{\beta}^b_j x^b_{ij} + \hat{\beta}^b_{j+1} x^b_{i(j+1)} + ... + \hat{\beta}^b_{k-1} x^b_{i(k-1)} = \hat{\beta}^b_j (x_{i(j+1)} - x_{ij} + x_{i(j+2)} - x_{i(j+1)} + ... + x_{ik} - x_{i(k-1)}) = \hat{\beta}^b_j (x_{ik} - x_{ij})$. Therefore the difference between x_{ij} and x_{ik} may be informative for classification. In such a case, the fused Lasso LR learns a **slope pattern**. An example is shown in Figure 2(b) where the difference between the values at time index 50 and 100 is informative to distinguish the two time series.

If $\hat{\beta}^c_j = \hat{\beta}^c_{j+1} = ... = \hat{\beta}^c_{k-1} \neq 0$, then $\hat{\beta}^c_j x^c_{ij} + \hat{\beta}^c_{j+1} x^c_{i(j+1)} + ... + \hat{\beta}^c_{k-1} x^c_{i(k-1)} = \hat{\beta}^c_j (|x_{i(j+1)} - x_{ij}| + ... + |x_{ik} - x_{i(k-1)}|)$. Therefore, the deviation of the values in the interval between time j and time k may be informative for classification. In other words, the deviation is the sum of the absolute differences of the consecutive values between time j and time k. In such a case, the fused Lasso LR learns a **deviation pattern**. An example is shown in Figure 2(c) where the deviation over the interval between time index 70 and 110 is informative to distinguish the two time series.

The original fused Lasso LR model only captures mean patterns. The new matrices X^b and X^c generated here allow additional patterns (slopes and deviations) to be easily incorporated into SMT. Others patterns may be added, or only one specific matrix may be used if the corresponding pattern is of interest. In addition, instead of splitting based on only one matrix from X^a, X^b and X^c, an alternative method is to split based on a larger matrix consisting of the concatenation of all three matrices $\tilde{X} = [X^a, X^b, X^c]$. We experimented with this setting, but did not detect improved accuracy between the use of $\tilde{X}$ and the matrices separately. Features from $\tilde{X}$ might also be a little more difficult to interpret.

4.3 Computational complexity

Let n^k_j denote the number of instances in the j^{th} node at the k^{th} depth in a SMT. As the total number of instances at each depth remains the same, $\sum_j n^k_j = N$. At each node, the solver [37] with the default settings has a computational complexity $O(n^k_j p)$ for solving the L_1 regularized LR models. Also, Step 8 from Algorithm 1 or Step 9 from Algorithm 2 has a complexity $O(n^k_j \log n^k_j)$ for ordering the candidate splitting points and selecting the splitting point with the best information gain. Here we consider the entropy splitting function as a worst case because ordering is not needed to split with $V = 0$. Therefore, the complexity at each depth is $O(\sum_j n^k_j p + \sum_j n^k_j \log n^k_j) \leq O(Np + \sum_j n^k_j \log N) = O(Np + N \log N)$. Denote the maximum depth of a SMT as K, then the computational complexity of SMT is $O(KN(p + \log N))$. Assuming that the maximum depth of a tree model to be $O(\log N)$ [39], the complexity becomes $O(N \log N \max[p, \log N])$.

5 Experiments

We tested SMT on the UCR time series data sets [38], the UCI data sets [42], and a digit classification problem [43]. Furthermore, SMT is tested on 13 sparse datasets with substantially more features than those in the UCI repository. The sparse datasets are more challenging and illustrate SMT's advantages in such tasks not only in performance, but also in computational efficiency.

We implemented the SMT models in Matlab. A sparse learning solver [37] was used to solve the Lasso LR and fused Lasso LR problems. Because the solver focuses on two-class models, only data with two classes were analyzed. The experiments were conducted on a 8 GB RAM, dual core CPU (i7-3620M 2.7 GHz) laptop.

5.1 Time series data

We tested SMT with fused Lasso LR on all binary classification problems from time series benchmark data sets from the UCR time series database [38]. The characteristics of the data sets are shown in Table 1. We also compared SMT to fused Lasso LR, Lasso LR, QUEST [15], CRUISE [18], two versions of LOTUS [21], LOTUS using simple logistic regression (Lotus(S)) and LOTUS using multiple logistic regression (Lotus(M)), FT [20], LMT [22], NBTree [19], 1-nearest-neighbor with Euclidean distance (NNEuclidean), and two versions of 1-nearest-neighbor with dynamic time warping: NNDTWBest [23] and NNDTWNoWin [38]. The results for the three NN classifiers were obtained by [38]. For SMT, fused Lasso LR and Lasso LR, each time series was standardized to zero mean and unit standard deviation.

Both λ_1 and λ_2 of SMT, Lasso LR and fused Lasso LR were chosen from $\{0.05, 0.1, 0.3, 0.5\}$ with the best error rate from five-fold cross-validation over the training data. Separate test data were used to evaluate the models. The sensitivity of SMT to λ_1 and λ_2 is compared to fused Lasso LR in Section 5.4.

The classification error rates on test data sets are shown in Table 2. A Wilcoxon matched-pairs signed-ranks test is used to compare SMT to competitor algorithms. Significance levels are shown for the alternative hypothesis that error rates from SMT are lower than competitors. Also, the counts of data sets where SMT accuracy is better, equal to, or worse than each competitor are provided. SMT is better than all competitors except NN classifiers at a 0.1 significance level. SMT is comparable to the two versions of NNDTW, which are considered strong solutions for time series problems [25]. For the instance-based classifiers, the computational complexity for testing is the same as training. However, SMT can quickly assign test data based on the pre-built tree. More importantly, instance-based classifiers can be difficult to interpret, but with SMT interpretable patterns described in Section 4.2 can be extracted. Here we describe the patterns of three datasets: GunPoint, Coffee and SonyAIBORobot Surface. Figures 3(a)-3(c) show example time series from each class, and the number of times a coefficient is nonzero in the nodes of the SMT for each data set. One notices the sparseness of the models in general, and that different locations in the time series tend to be summarized with either mean, slope, or deviation features.

GunPoint: GunPoint dataset is one of the most studied time series classification problem [44]. The aim is to classify a motion as 'Gun' or 'NoGun' through time series generated by mapping the motion of two actors. For the Gun class, the actors draw a gun from a holster, aim at a target, and return the gun to the holster [45]. In the NoGun class, actors perform the same movements without a gun. Instead, they use their index finger to point to a target. Therefore, in NoGun class, the step of drawing the gun from a holster and returning it back is skipped. Figure 3 illustrates the mean and deviation features from the regions related to the actions, "draw a gun from a hip-mounted holster" and "return the gun to the holster and their hands to their sides". The discriminative actions are well captured by the mean and deviation patterns.

Table 1: Characteristics of the time series data sets: the number of training and test instances, and the time series lengths.

	Training Instances	Testing Instances	Length
Coffee	28	28	286
ECG200	100	100	96
ECGFiveDays	23	861	136
GunPoint	50	150	150
ItalyPowerDemand	67	1029	24
Lighting2	60	61	637
MoteStrain	20	1252	84
SonyAIBORobot	20	601	70
SonyAIBORobotII	27	953	65
TwoLeadECG	23	1139	82
Wafer	1000	6174	152
Yoga	300	3000	426

Coffee: The task is to classify the coffee species in instant coffees. A chemical analysis, called Diffuse Reflectance Infrared Fourier Transform (DRIFT), is used to discriminate between two species of coffees as Arabica and Robusta [46]. The error rates of NNBestDTW and NNDTWNoWin are 0.179. Also, [46] stated that certain spectral regions represent the caffeine bands. These regions correspond to the time frame between 187.7 and 247.3 as discussed by [45]. SMT is able to extract the temporal features from these regions, as illustrated in Figure 3.

Sony AIBO Robot: This dataset was created by [47] and the task is to classify the surface types using the measurements of the tri-axial accelerometer from Sony AIBO Robot [48]. Only the x-axis readings of the accelerometer were provided by [38]. Two types of surfaces, carpet and cement, are considered in this dataset. Cement floors are harder, resulting in sharper changes in the acceleration [48]. The error rates of NNBestDTW and NNDTWNoWin are 0.305 and 0.275, respectively. SMT is substantially more accurate than NN. The important patterns provided in Figure 3(c) are similar to the shapelets by [48] and they refer to different shifts-of-weight in the walk cycle on the carpet floor.

5.2 UCI data

We tested SMT with Lasso LR on the data sets from the UCI database [42] and a 3-versus-8 digit classification problem from [43]. Table 3 shows the data characteristics. The datasets are sorted based on the total number of features in descending order. We compared SMT to Lasso LR, QUEST [15], CRUISE [18], two ver-

Table 2: Error rates from SMT(IG), SMT(V=0) and other classifiers for time series classification. Significance levels from a Wilcoxon matched-pairs signed-ranks test for the alternative hypothesis that error rates from SMT(IG) are lower than competitors are provided.

	SMT(IG)	SMT (V=0)	Fused Lasso LR	Lasso LR	LMT	LOTUS(S)	LOTUS(M)
Coffee	0.036	0.036	0.036	0.071	0.000	0.536	0.536
ECG200	0.170	0.170	0.190	0.170	0.180	0.200	0.200
ECGFiveDays	0.158	0.101	0.091	0.066	0.335	0.214	0.231
GunPoint	0.127	0.167	0.260	0.220	0.207	0.307	0.480
ItalyPowerDemand	0.039	0.040	0.038	0.030	0.030	0.030	0.030
Lighting2	0.295	0.344	0.328	0.459	0.361	0.279	0.328
MoteStrain	0.133	0.168	0.168	0.124	0.203	0.197	0.197
SonyAIBORobot	0.130	0.225	0.311	0.225	0.283	0.571	0.266
SonyAIBORobotII	0.200	0.196	0.217	0.217	0.219	0.383	0.226
TwoLeadECG	0.221	0.208	0.218	0.145	0.114	0.315	0.315
Wafer	0.011	0.008	0.070	0.070	0.019	0.014	0.064
Yoga	0.206	0.274	0.366	0.349	0.281	0.464	0.464
p-value	-	-	0.028	0.071	0.046	0.004	0.002
win/lose/tie	-	-	3/8/1	4/7/1	3/9/0	2/10/0	1/11/0

	C4.5	NBTree	FT	QUEST	CRUISE	NNEuclidean	NNBestDTW	NNDTWNoWin
Coffee	0.429	0.357	0.464	0.000	0.000	0.250	0.179	0.179
ECG200	0.280	0.230	0.210	0.360	0.360	0.120	0.120	0.230
ECGFiveDays	0.279	0.220	0.503	0.323	0.503	0.203	0.203	0.232
GunPoint	0.227	0.247	0.213	0.180	0.193	0.120	0.087	0.093
ItalyPowerDemand	0.053	0.064	0.028	0.073	0.064	0.045	0.045	0.050
Lighting2	0.377	0.312	0.295	0.459	0.459	0.246	0.131	0.131
MoteStrain	0.213	0.142	0.461	0.133	0.276	0.121	0.134	0.165
SonyAIBORobot	0.344	0.260	0.571	0.571	0.296	0.305	0.305	0.275
SonyAIBORobotII	0.314	0.210	0.383	0.195	0.192	0.141	0.141	0.169
TwoLeadECG	0.282	0.268	0.500	0.091	0.111	0.253	0.132	0.096
Wafer	0.021	0.008	0.025	0.037	0.037	0.005	0.005	0.050
Yoga	0.301	0.266	0.315	0.441	0.406	0.170	0.155	0.164
p-value	0.001	0.002	0.002	0.098	0.015	0.578	0.748	0.333
win/lose/tie	0/12/0	1/11/0	1/11/0	3/8/1	3/9/0	7/5/0	7/5/0	5/7/0

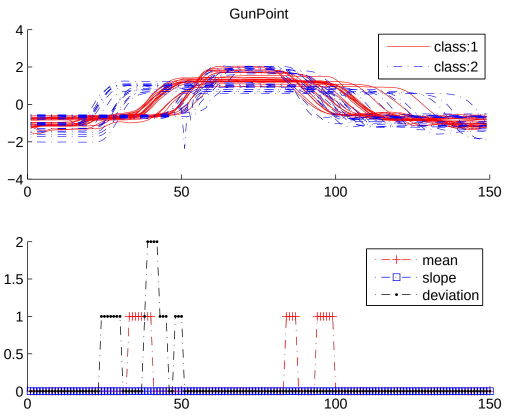
Table 3: Characteristics of the UCI datasets sorted by number of features (descending).

Dataset	Instances	% Missing	Numeric	Nominal	Ordinal
Madelon	200	0	500	0	0
zip3vs8	1200	0	256	0	0
Musk	476	0	168	0	0
Sonar	208	0	60	0	0
Ionosphere	351	0	34	0	0
Sick	3772	29.9	6	20	0
Annthyroid	7200	0	6	15	0
German	1000	0	7	13	0
Hepatitis	155	5.6	6	13	0
Labor	57	35.7	8	0	8
Letter-A	20000	0	16	0	0
Pendigit	10992	0	16	0	0
Australian credit	690	5.4	6	9	0
Boston	506	0	12	1	0
Heart	303	2	6	7	0
Heart-statlog	270	0	13	0	0
Pageblock	5473	0	10	0	0
Breast	699	2.3	9	0	0
Contracep	1473	0	2	4	3
Glass (G2)	163	0	9	0	0
Abalone	4177	0	7	1	0
California	20640	0	8	0	0
Pima	768	0	8	0	0
Yeast	1484	0	8	0	0
Bupa	345	0	6	0	0

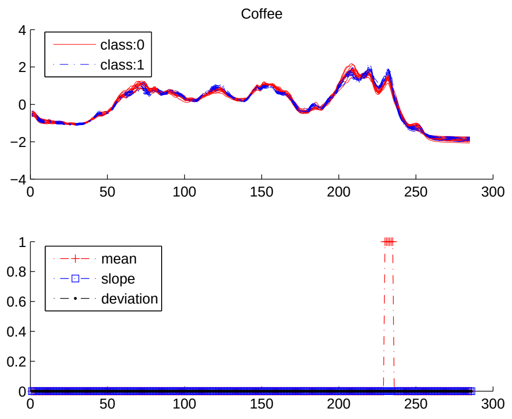
sions of LOTUS [21], LOTUS using simple logistic regression (Lotus(S)) and LOTUS using multiple logistic regression (Lotus(M)), FT [20], LMT [22], and NBTree [19]. To investigate the effect of the Lasso LR penalty, we set λ_1 to an extremely small value (such as 10^{-50}) because the solver does not allow for a zero assignment. This model is referred to as a multivariate logistic regression tree (MLRT). For SMT, Lasso LR and MLRT, each attribute was standardized to zero mean and unit standardization.

For every dataset and algorithm, we performed 10 runs of five-fold cross-validation to evaluate the performance (using the same splits into training/test set for every method). This yields 50 error rates for each algorithm and dataset. Another five-fold cross-validation was nested within each training fold in which λ_1 of SMT and Lasso LR were chosen from {0.05, 0.1, 0.3, 0.5} based on the best average error. The sensitivity of SMT to λ_1 is compared to Lasso LR in Section 5.4. The mean and the standard deviation of the classification error rates over the 10 replicates are shown in Tables 4 and 5. We also performed a paired t-test to compare the algorithms for each dataset. For each dataset, algorithms performing significantly better or worse than SMT (at a 0.1 significance level) are indicated with an open or filled circle, respectively, in the adjacent column. The number of algorithms that perform significantly worse than SMT at a 0.1 significance level (wins) is provided in the last column, along with the significant losses and ties.

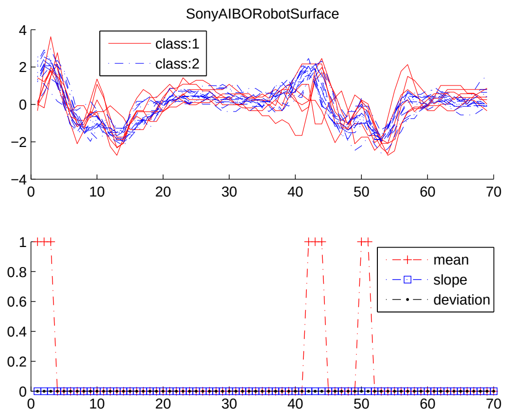
The datasets are listed in order (descending) of the number of features and the filled circles indicate that SMT tends to perform better for datasets with a greater



(a) Gun point: mean and deviation patterns are effective for classification.



(b) Coffee: mean patterns are effective for classification.



(c) Sony AIBO Robot: mean patterns are effective for classification.

Figure 3: Each figure shows example time series from each class and the number of times a coefficient is nonzero in the nodes of the SMT for each dataset. Here -+-, -□-, -●- represent the mean, slope, and deviation patterns, respectively.

number of features. Other methods outperform MLRT and SMT (at a 0.1 significance level) in datasets with a small number of features. However, the differences in the average error rates are not large. In particular, LMT is a strong competitor to SMT in general for non-time-series data, but not for time series data. Furthermore, as shown later, LMT has substantially greater computational requirements.

5.3 Sparse data

We introduce 13 additional datasets from NIPS 2003 feature selection challenge [49], WCCI 2006 performance prediction challenge [50] and the feature selection Website from [51] to illustrate the advantages of our approach against the stronger competitors (from the previous results).

Table 6 summarizes the characteristics of the datasets and the results for stronger competitors of SMT. The problems associated with computational complexity arise for other approaches in large datasets. We ran the competitor algorithms on more powerful hardware because of their computational requirements. For some datasets, some of the algorithms did not return a result in 48 hours or they terminated because of the memory limitations (on 32 GB RAM server). We represent these with an "x" in Table 6. SMT is run on the same configuration described in Section 5 (a laptop with 8 GB RAM). We ran five replicates of five-fold cross-validation for each dataset because of the requirements of the competitors in terms of computation time. This yields 25 error rates for each algorithm and dataset. The mean and the stan-

Table 4: Error rates from SMT and logistic regression based classifiers for UCI datasets. SMT results are better for datasets with more features near the top of the table.

	SMT (V=0)	MLRT (V=0)		SMT (IG)	MLRT (IG)	Lasso LR		LMT		LOTUS(S)		LOTUS(M)		w/l/t*
Madelon	0.359±0.007	0.482±0.011	●	0.384±0.016	0.474±0.014	0.387±0.005	●	0.349±0.007	○	0.432±0.025	●	0.413±0.003	●	4/1/0
zip3vs8	0.031±0.004	0.021±0.004	○	0.030±0.004	0.020±0.002	0.036±0.004	●	0.025±0.005	○	0.062±0.009	●	0.043±0.005	●	3/2/0
Musk	0.180±0.010	0.158±0.015	○	0.160±0.021	0.163±0.009	0.190±0.011	●	0.164±0.016	○	0.299±0.036	●	0.214±0.011	●	3/2/0
Sonar	0.240±0.027	0.255±0.029		0.219±0.008	0.259±0.015	0.256±0.022	●	0.244±0.007		0.291±0.018	●	0.276±0.018	●	3/0/2
Ionosphere	0.102±0.012	0.153±0.013	●	0.111±0.011	0.158±0.006	0.127±0.010	●	0.090±0.015	○	0.105±0.008		0.217±0.016	●	3/1/1
Sick	0.036±0.002	0.033±0.002	○	0.019±0.001	0.036±0.003	0.038±0.002	●	0.011±0.002	○	0.026±0.002	○	0.028±0.002	○	1/4/0
Annthyroid	0.017±0.001	0.015±0.002	○	0.011±0.002	0.022±0.002	0.050±0.001	●	0.006±0.001	○	0.035±0.001	●	0.030±0.001	●	3/2/0
German	0.258±0.007	0.274±0.013	●	0.318±0.007	0.319±0.014	0.250±0.008	○	0.253±0.005	○	0.288±0.008	●	0.262±0.008		2/2/1
Hepatitis	0.170±0.026	0.203±0.025	●	0.190±0.015	0.215±0.025	0.156±0.011	○	0.161±0.008		0.223±0.011	●	0.195±0.027	●	3/1/1
Labor	0.108±0.030	0.072±0.016	○	0.117±0.021	0.107±0.032	0.113±0.023		0.142±0.023	●	0.212±0.016	●	0.348±0.029	●	3/1/1
Letter-A	0.008±0.001	0.007±0.001	○	0.006±0.001	0.010±0.001	0.011±0.001	●	0.003±0.000	○	0.010±0.001	●	0.006±0.001	○	2/3/0
Pendigit	0.015±0.001	0.012±0.001	○	0.003±0.001	0.006±0.001	0.018±0.001	●	0.003±0.001	○	0.007±0.001	○	0.004±0.001	○	1/4/0
Australian credit	0.150±0.004	0.172±0.007	●	0.188±0.009	0.212±0.008	0.145±0.003	○	0.144±0.006	○	0.150±0.005		0.149±0.007		1/2/2
Boston	0.133±0.007	0.137±0.009		0.165±0.010	0.177±0.016	0.137±0.009	●	0.139±0.009	●	0.152±0.009	●	0.138±0.006	●	4/0/1
Heart	0.180±0.012	0.201±0.013	●	0.249±0.012	0.257±0.023	0.171±0.007	○	0.180±0.006		0.206±0.015	●	0.185±0.008		2/1/2
Heart-statlog	0.176±0.011	0.200±0.015	●	0.241±0.009	0.260±0.025	0.166±0.010	○	0.168±0.007	○	0.217±0.009	●	0.185±0.006	●	3/2/0
Pageblock	0.044±0.001	0.042±0.002	○	0.035±0.003	0.039±0.003	0.056±0.001	●	0.029±0.002	○	0.034±0.003	○	0.031±0.002	○	1/4/0
Breast	0.039±0.004	0.040±0.004		0.054±0.010	0.059±0.006	0.042±0.003	●	0.039±0.005		0.053±0.006	●	0.040±0.004		2/0/3
Contracep	0.320±0.005	0.322±0.004		0.372±0.014	0.384±0.012	0.327±0.004	●	0.302±0.003	○	0.304±0.005	○	0.288±0.004	○	1/3/1
Glass (G2)	0.268±0.024	0.323±0.028	●	0.263±0.020	0.303±0.037	0.274±0.024		0.235±0.023	○	0.260±0.024		0.252±0.013	○	1/2/2
Abalone	0.228±0.003	0.220±0.003	○	0.283±0.005	0.290±0.009	0.241±0.002	●	0.213±0.003	○	0.240±0.006	●	0.212±0.002	○	2/3/0
California	0.147±0.001	0.136±0.001	○	0.154±0.011	0.178±0.009	0.155±0.001	●	0.109±0.012	○	0.104±0.008	○	0.099±0.005	○	1/4/0
Pima	0.240±0.008	0.240±0.009		0.298±0.025	0.330±0.014	0.236±0.004	○	0.230±0.005	○	0.251±0.007	●	0.236±0.004	○	1/3/1
Yeast	0.102±0.003	0.105±0.003	●	0.133±0.005	0.144±0.006	0.106±0.003	●	0.089±0.002	○	0.092±0.003	○	0.089±0.005	○	2/3/0
Bupa	0.336±0.014	0.333±0.014		0.362±0.015	0.386±0.025	0.325±0.015	○	0.305±0.019	○	0.367±0.017	●	0.321±0.016	○	1/3/1

*(win/lose/tie) Compare SMT (V=0) with MLRT (V=0), Lasso LR, LMT, LOTUS(S), LOTUS(M) ●, ○ statistically significant win or loss for SMT (V=0)

Table 5: Error rates from SMT and other classifiers for UCI datasets.

	SMT (V=0)	C4.5		FT		NBTree		QUEST		CRUISE		w/l/t
Madelon	0.359±0.007	0.358±0.005		0.327±0.012	○	0.456±0.025	●	0.463±0.031	●	0.465±0.034	●	3/1/1
zip3vs8	0.031±0.004	0.054±0.005	●	0.033±0.003	●	0.048±0.005	●	0.036±0.003	●	0.038±0.005	●	5/0/0
Musk	0.180±0.010	0.194±0.028	●	0.185±0.015		0.182±0.021		0.201±0.014	●	0.190±0.010	●	3/0/2
Sonar	0.240±0.027	0.290±0.023	●	0.235±0.019		0.271±0.018	●	0.266±0.024	●	0.268±0.023	●	4/0/1
Ionosphere	0.102±0.012	0.103±0.017		0.110±0.009	●	0.099±0.013		0.097±0.007	○	0.109±0.013		1/1/3
Sick	0.036±0.002	0.014±0.001	○	0.021±0.002	○	0.020±0.003	○	0.034±0.001	○	0.035±0.002	○	0/5/0
Annthyroid	0.017±0.001	0.005±0.001	○	0.008±0.001	○	0.006±0.001	○	0.011±0.001	○	0.018±0.001	●	1/4/0
German	0.258±0.007	0.293±0.012	●	0.295±0.007	●	0.288±0.007	●	0.255±0.004	○	0.262±0.005		3/1/1
Hepatitis	0.170±0.026	0.239±0.011	●	0.184±0.022	●	0.183±0.023		0.176±0.011		0.203±0.008	●	3/0/2
Labor	0.108±0.030	0.145±0.024	●	0.118±0.021		0.152±0.032	●	0.139±0.030	●	0.125±0.021	●	4/0/1
Letter-A	0.008±0.001	0.005±0.001	○	0.006±0.001	○	0.006±0.001	○	0.006±0.001	○	0.008±0.002	○	0/5/0
Pendigit	0.015±0.001	0.004±0.001	○	0.005±0.001	○	0.004±0.001	○	0.041±0.002	●	0.039±0.001	●	2/3/0
Australian credit	0.150±0.004	0.157±0.006	●	0.163±0.006	●	0.166±0.007	●	0.148±0.003	○	0.143±0.003	○	3/2/0
Boston	0.133±0.007	0.162±0.011	●	0.136±0.011		0.149±0.011	●	0.147±0.007	●	0.149±0.006	●	4/0/1
Heart	0.180±0.012	0.236±0.012	●	0.195±0.011	●	0.211±0.028	●	0.171±0.006	○	0.168±0.007	○	3/2/0
Heart-statlog	0.176±0.011	0.209±0.009	●	0.193±0.011	●	0.214±0.014	●	0.158±0.006	○	0.167±0.003	○	3/2/0
Pageblock	0.044±0.001	0.029±0.003	○	0.030±0.002	○	0.031±0.002	○	0.040±0.001	○	0.040±0.002	○	0/5/0
Breast	0.039±0.004	0.058±0.005	●	0.035±0.005	○	0.038±0.004		0.046±0.003	●	0.032±0.002	○	2/2/1
Contracep	0.320±0.005	0.312±0.005	○	0.299±0.005	○	0.311±0.003	○	0.318±0.006		0.324±0.004	●	1/3/1
Glass (G2)	0.268±0.024	0.216±0.021	○	0.274±0.021		0.195±0.012	○	0.313±0.011	●	0.305±0.017	●	2/2/1
Abalone	0.228±0.003	0.244±0.006	●	0.226±0.006		0.272±0.008	●	0.217±0.003	○	0.220±0.004	○	2/2/1
California	0.147±0.001	0.107±0.009	○	0.114±0.007	○	0.126±0.008	○	0.139±0.009	○	0.130±0.008	○	0/5/0
Pima	0.240±0.008	0.261±0.015	●	0.236±0.022		0.258±0.011	●	0.232±0.005	○	0.246±0.005	●	3/1/1
Yeast	0.102±0.003	0.094±0.004	○	0.093±0.002	○	0.096±0.003	○	0.103±0.002	●	0.103±0.003		1/3/1
Bupa	0.336±0.014	0.337±0.035		0.355±0.013	●	0.310±0.017	○	0.317±0.011	○	0.346±0.017		1/2/2

*(win/lose/tie) Compare SMT (V=0) with C4.5, FT, NBTree, QUEST, CRUISE. ●, ○ statistically significant win or loss for SMT (V=0)

Table 6: Error rates from SMT and other classifiers on sparse problems. Datasets sorted by sparseness (ratio p/N) and better results occur for SMT in the more sparse datasets (those above the horizontal line).

Dataset	Features	Instances	p/N	SMT (V=0)	LMT		FT		CRUISE		QUEST		LOTUS (S)		LOTUS (M)	
GLI-85	22283	85	262.2	0.125±0.066	0.161±0.079	•	0.157±0.065	•	x		0.169±0.159		x		0.157±0.057	•
Dorothea	100000	800	125	0.073±0.016	x		x		x		x		x		x	
SMK-CAN-187	19993	187	106.9	0.292±0.044	0.308±0.058		0.321±0.061	•	x		0.281±0.051		x		0.385±0.097	•
Arcene	10000	100	100	0.250±0.068	0.289±0.080	•	0.279±0.052	•	x		0.106±0.049	○	x		0.303±0.083	•
Dexter	20000	300	66.7	0.070±0.026	0.095±0.023	•	0.087±0.029	•	x		0.068±0.010		x		0.205±0.031	•
NOVA	16969	1754	9.7	0.097±0.016	x		0.103±0.013		x		0.110±0.017	•	x		0.151±0.034	•
BASEHOCK	4862	1993	2.4	0.040±0.013	0.043±0.012		0.058±0.010	•	x		0.061±0.013	•	0.202±0.012	•	0.190±0.011	•
PCMAC	3289	1943	1.7	0.095±0.018	0.087±0.020	○	0.115±0.020	•	x		0.224±0.020	•	0.215±0.018	•	0.228±0.028	•
Gisette	5000	6000	0.8	0.035±0.005	x		0.049±0.006	•	x		x		x		x	
HIVA	1617	3845	0.4	0.032±0.008	0.031±0.007	○	0.047±0.009	•	0.039±0.013	•	0.036±0.009	•	0.041±0.012	•	0.046±0.012	•
GINA	970	3153	0.3	0.125±0.013	0.104±0.010	○	0.152±0.012	•	0.191±0.016	•	0.195±0.014	•	0.137±0.015	•	0.145±0.010	•
SYLVA	216	13086	0	0.009±0.002	0.008±0.001	○	0.009±0.002		0.008±0.002		0.010±0.002		0.013±0.004	•	0.016±0.003	•
ADA	48	4147	0	0.152±0.013	0.145±0.010	○	0.159±0.013	•	0.152±0.012		0.161±0.014	•	0.166±0.015	•	0.156±0.011	

dard deviation of the classification error rates over the five replicates are shown in Table 6. For each dataset, algorithms performing significantly better or worse than SMT (at a 0.1 significance level) are indicated with an open or filled circle, respectively, in the adjacent column. Datasets are sorted by sparseness (ratio p/N) to illustrate the advantages of our approach. Better results occur for SMT in the more sparse datasets. Only QUEST performs equally well for the datasets above the horizontal line, but the computational requirements of QUEST are major problems. Also, SMT outperforms QUEST for less sparse datasets in Table 6.

5.4 Sensitivity analysis

Here we investigate the sensitivity of SMT to the changes of the parameters: λ_1 and λ_2 (only for time series) with $\lambda_1, \lambda_2 \in \{0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$. The training/testing split setting and parameter selection are the same as Section 5.1. For time series, only the results of the yoga data are shown in Figure 4, but the other time series have similar results. The results for the non-time-series data sets are shown in Figure 5. For most time series and non-time-series data sets, the error rates from SMT are much less sensitive to λ_1 than Lasso LR and fused Lasso LR. As λ_1 increases, the number of features used in Lasso LR and fused Lasso LR, and the average number of features used in a SMT non-leaf node decrease quickly. Also, as λ_1 increases, the depth of SMT initially increases and then tends to level off or decrease. The objective 2 favors sparsity with larger λ_1 values, and the empirical loss (first term) becomes less important. Therefore, small information gain from splitting based on $V = 0$ is expected after a certain depth. Similarly, the increase in error rates can be observed near the same values where depth starts to level off or decrease. Focusing more on the sparsity than the empirical loss tends to increases the error rate. For time series, both the error rates and the number of features are not overly sensitive to λ_2 for both fused Lasso LR and SMT.

5.5 Computational time analysis

Here we empirically evaluate the runtime to train a SMT model as either the number of features or the number of training instances changes. SMT with Lasso LR ($\lambda_1 = 0.1$) was used for non-time-series data sets. SMT with fused Lasso LR ($\lambda_1 = 0.1$, $\lambda_2 = 0.1$) was used for the time series data sets. For each data set, we randomly selected $\delta \in \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$ proportion of either instances or features. Here 10 replicates were conducted for each setting, and the average runtime for each data set is shown in Figure 6.

Figures 6(a) and 6(b) show the runtime for time series data sets as the number of features and instances change, respectively. Figures 6(c) and 6(d) show similar results for non-time-series data sets. For most data sets, the runtime increase with the number of features is not obvious. The runtime increase with the number instances is more obvious. This is consistent with the complexity $O(n \log n \max[p, \log n])$ mentioned in Section 4.3. That is, the number of instances has a larger impact on runtime than the number of features. In some cases, the runtime decreased marginally with the number features or instances. This may indicate the influence of the data characteristics such as relatively easier classification problems, or problems with more informative features/instances that require less time to learn (e.g. a SMT may have only depth one for an easy classification problem). Figures 6(e) and 6(f) show the runtime for LMT with the number of features and instances, respectively. The LMT runtimes are much greater than SMT for the same number of features or instances. Runtime of LMT increases linearly as the number of features increases since LMT iteratively fits simple linear regression functions to find the best set of features. [22] also states that fitting the logistic regression models with the LogitBoost [52] at each node of the tree makes the algorithm slow in practice.

6 Conclusions

Univariate trees such as CART [1] are limited to using only one variable to split a node. Multivariate trees use a function of multiple variables for splitting, and, thus, can be more expressive than univariate trees. However, a multivariate tree using many variables at each node can be difficult to understand.

To this end, a sparse multivariate tree (SMT) framework is presented that applies easily to both non-time-series and time-series classification problems. The objective is to handle sparse problems with the help of regularization and recursive partitioning enabled by trees. Handling problems with a large number of features and few instances is a challenge because the curse of dimensionality is acute and there are insufficient degrees of freedom to estimate the full model. Furthermore, the SMT models for time-series data enables one to extract simple temporal patterns (means, slopes, deviations, or others) through a simple transform of the data matrix within the framework.

Extensive experiments have been conducted to investigate the performance of SMT relative to a large number of competitors. For non-time-series problems, SMT does not have an advantage on data sets with a relatively small number of features. However, SMT is more accurate than other methods when the number of features is much larger than the number of instances (sparse problems). For time-series problems, SMT has better accuracy than other tree-based models. SMT has similar accuracy compared to strong time-series classifiers such as nearest neighbor classifiers with dynamic time warping, but SMT can extract temporal features that are useful for classification, and, thus, is more interpretable than NN classifiers. SMT is also computationally efficient with regard to both the number of features and the number of instances.

In this work, binary L_1 regularized logistic regression models are used in SMT for binary classification problems. However, SMT may be extended to solve multi-class problems by using multinomial logistic regression models at tree nodes. Furthermore, it is well-known that an ensemble model consisting of multiple base classifiers is, in general, more accurate than one single base classifier. Consequently, it may be valuable to develop a mechanism to combine multiple SMT trees for better accuracy. Such an ensemble model can be less interpretable. However, one might obtain insights into the temporal features by aggregating the information from all the trees [11].

References

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

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

[3] C. Brodley and P. Utgoff, "Multivariate decision trees," *Machine Learning*, vol. 19, no. 1, pp. 45–77, 1995.

[4] J. Gama, "Oblique linear tree," *Advances in Intelligent Data Analysis Reasoning about Data*, pp. 187–198, 1997.

[5] X. Li, J. Sweigart, J. Teng, J. Donohue, L. Thombs, and S. Wang, "Multivariate decision trees using linear discriminants and tabu search," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 33, no. 2, pp. 194–205, 2003.

[6] D. Eads, K. Glocer, S. Perkins, and J. Theiler, "Grammar-guided feature extraction for time series classification," in *Proceedings of Conference on Neural Information Processing Systems (NIPS05)*, 2005.

[7] 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.

[8] 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.

[9] 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.

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

[11] H. Deng, G. Runger, E. Tuv, and M. Vladimir, "A time series forest for classification and feature extraction," *Information Sciences*, vol. 239, pp. 142–153, Aug. 2013.

[12] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight, "Sparsity and smoothness via the fused lasso," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 1, pp. 91–108, 2005.

[13] S. K. Murthy, S. Kasif, and S. Salzberg, “A system for induction of oblique decision trees,” *Journal of Artificial Intelligence Research*, vol. 2, pp. 1–32, 1994.

[14] E. Cantu-Paz and C. Kamath, “Inducing oblique decision trees with evolutionary algorithms,” *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 1, pp. 54–68, 2003.

[15] W. Loh and Y. Shih, “Split selection methods for classification trees,” *Statistica Sinica*, vol. 7, pp. 815–840, 1997.

[16] X. Li, “A scalable decision tree system and its application in pattern recognition and intrusion detection,” *Decision Support Systems*, vol. 41, no. 1, pp. 112–130, 2005.

[17] O. Yildiz and E. Alpaydin, “Linear discriminant trees,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 19, no. 3, pp. 323–353, 2005.

[18] H. Kim and W. Loh, “Classification trees with unbiased multiway splits,” *Journal of the American Statistical Association*, vol. 96, no. 454, pp. 589–604, 2001.

[19] R. Kohavi, “Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid,” in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, vol. 7. Menlo Park, USA: AAAI Press, 1996.

[20] J. Gama, “Functional trees,” *Machine Learning*, vol. 55, no. 3, pp. 219–250, 2004.

[21] K. Y. Chan and W. Y. Loh, “Lotus: An algorithm for building accurate and comprehensible logistic regression trees,” *Journal of Computational and Graphical Statistics*, vol. 13, no. 4, pp. 826–852, 2004.

[22] N. Landwehr, M. Hall, and E. Frank, “Logistic model trees,” *Machine Learning*, vol. 59, no. 1, pp. 161–205, 2005.

[23] 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.

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

[25] C. Ratanamahatana and E. Keogh, “Three myths about dynamic time warping data mining,” in *Proceedings of SIAM International Conference on Data Mining (SDM05)*, vol. 21, 2005, pp. 506–510.

[26] L. Ye and E. Keogh, “Time series shapelets: a new primitive for data mining,” in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*. ACM, 2009, pp. 947–956.

[27] Z. Xing, J. Pei, P. S. Yu, and K. Wang, “Extracting interpretable features for early classification on time series,” in *Proceedings of SIAM International Conference on Data Mining (SDM)*. SIAM, 2011, pp. 247–258.

[28] J. Lines, L. M. Davis, J. Hills, and A. Bagnall, “A shapelet transform for time series classification,” in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*. ACM, 2012, pp. 289–297.

[29] 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.

[30] P. Geurts, “Pattern extraction for time-series classification,” in *Proceedings of PKDD 2001, 5th European Conference on Principles of Data Mining and Knowledge Discovery*, ser. LNAI 2168, L. de Raedt and A. Siebes, Eds. Freiburg: Springer-Verlag, September 2001, pp. 115–127.

[31] 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.

[32] 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.

[33] R. Isermann, “Supervision, fault-detection and fault-diagnosis methods an introduction,” *Control Engineering Practice*, vol. 5, no. 5, pp. 639 – 652, 1997.

[34] K. Koh, S. Kim, and S. Boyd, “An interior-point method for large-scale l1-regularized logistic regression,” *Journal of Machine Learning Research*, vol. 8, no. 8, pp. 1519–1555, 2007.

[35] R. Tibshirani, "Optimal reinsertion: Regression shrinkage and selection via the lasso," *J.R.Statist Soc. B*, vol. 58, no. 1, pp. 267–288, 1996.

[36] J. Liu, L. Yuan, and J. Ye, "An efficient algorithm for a class of fused lasso problems," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2010, pp. 323–332.

[37] J. Liu, S. Ji, and J. Ye, "Slep: Sparse learning with efficient projections," *Arizona State University*, 2009.

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

[39] I. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques.* Morgan Kaufmann, 2005.

[40] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explorations*, vol. 11, no. 1, pp. 10–18, 2009.

[41] M. Saar-Tsechansky and F. Provost, "Handling missing values when applying classification models," *J. Mach. Learn. Res.*, vol. 8, pp. 1623–1657, Dec. 2007.

[42] C. Blake and C. Merz, "Uci repository of machine learning databases," 1998.

[43] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, 2nd ed. Springer, 2009.

[44] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, "Querying and mining of time series data: experimental comparison of representations and distance measures," *Proc. VLDB Endow.*, vol. 1, pp. 1542–1552, August 2008.

[45] L. Ye and E. Keogh, "Time series shapelets: a novel technique that allows accurate, interpretable and fast classification," vol. 22. Springer Netherlands, 2011, pp. 149–182.

[46] R. Briandet, E. K. Kemsley, and R. H. Wilson, "Discrimination of arabica and robusta in instant coffee by fourier transform infrared spectroscopy and chemometrics," *Journal of Agricultural and Food Chemistry*, vol. 44, no. 1, pp. 170–174, 1996.

[47] D. Vail and M. Veloso, "Learning from accelerometer data on a legged robot," in *In Proceedings of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles*, 2004.

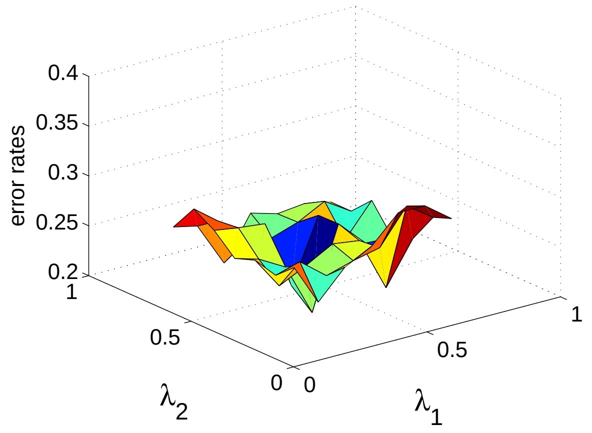
[48] A. Mueen, E. J. Keogh, and N. Young, "Logical-shapelets: an expressive primitive for time series classification." in *KDD.* ACM, 2011, pp. 1154–1162.

[49] I. Guyon, A. B. Hur, S. Gunn, and G. Dror, "Result analysis of the nips 2003 feature selection challenge," in *Advances in Neural Information Processing Systems 17.* MIT Press, 2004, pp. 545–552.

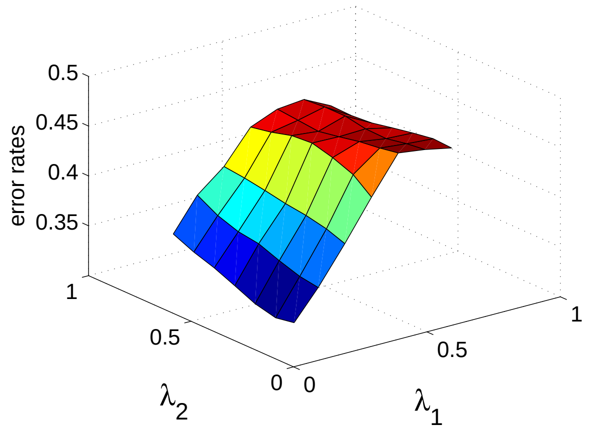
[50] I. Guyon, A. Alamdari, G. Dror, and J. Buhmann, "Performance prediction challenge," in *Neural Networks, 2006. IJCNN '06. International Joint Conference on*, 2006, pp. 1649–1656.

[51] F. S. at Arizona State University, "http://featureselection.asu.edu/," accessed: February 10, 2013.

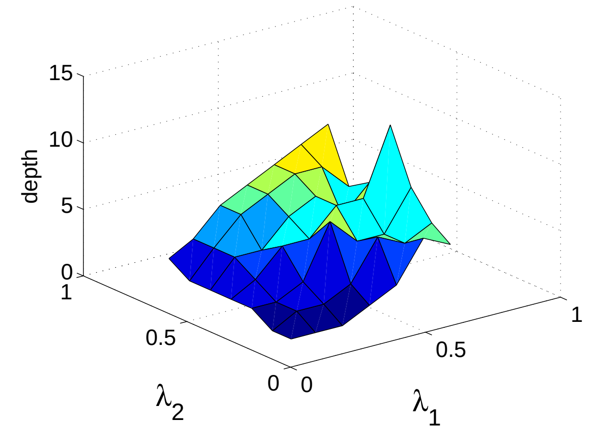
[52] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *Annals of Statistics*, vol. 28, p. 2000, 1998.



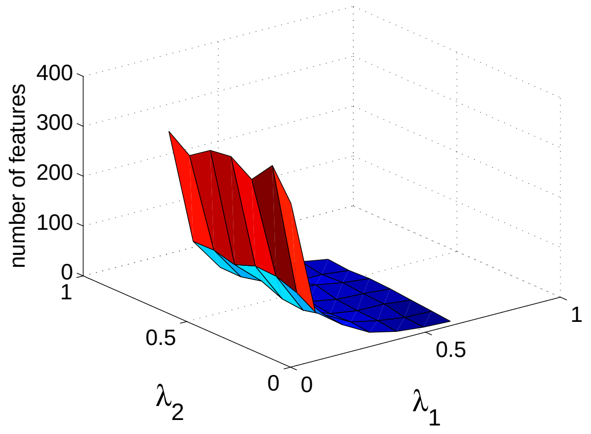
(a) The error rates of SMT



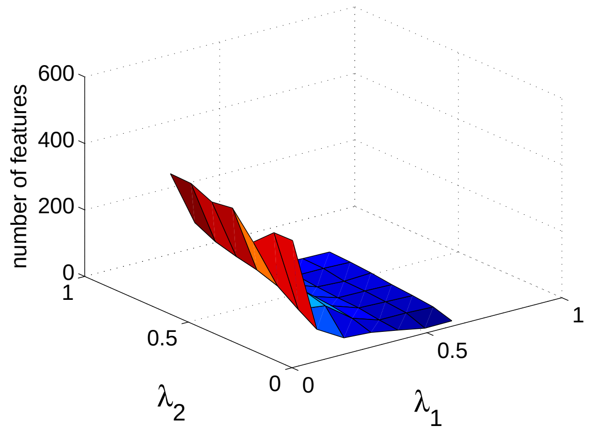
(b) The error rates of fused Lasso LR



(c) The depth of SMT

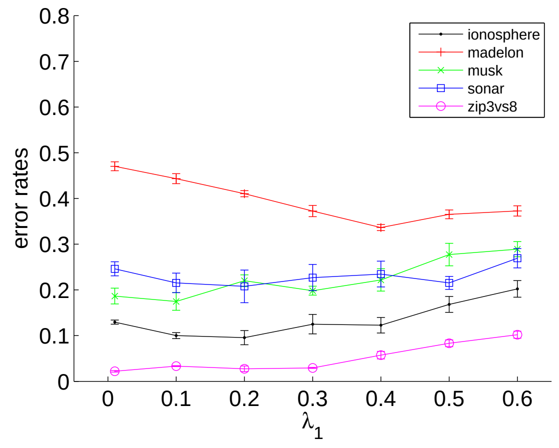


(d) The average number of features used in a node of SMT

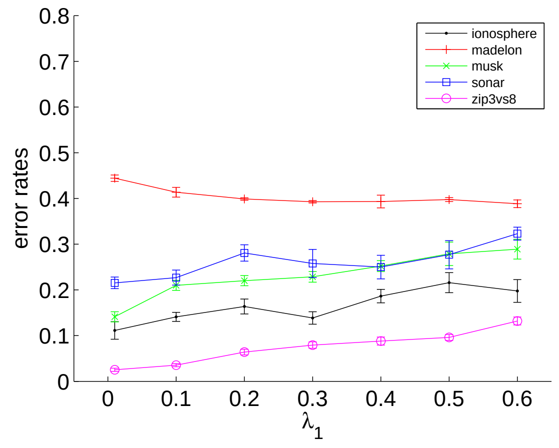


(e) The number of features used in fused Lasso LR

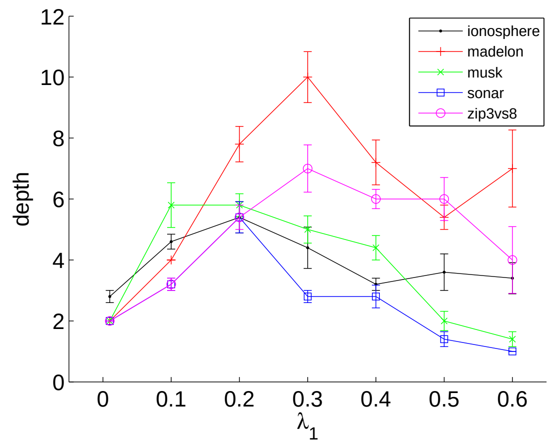
Figure 4: Sensitivity analysis of SMT and fused Lasso LR for the yoga time series data set. The above figures show the changes of the error rates, the number of features and the depth (only for SMT) due to the change of λ_1 and λ_2. Both the error rates and the number of features are not sensitive to λ_2. As λ_1 increases, the error rates from fused Lasso LR increase more obviously than SMT, and both the number of features used in fused Lasso LR and the average number of features used in a SMT non-leaf node decrease. As λ_1 increases, the depth of SMT also increases. However, the average number of features used in a non-leaf SMT node decreases at a faster rate.



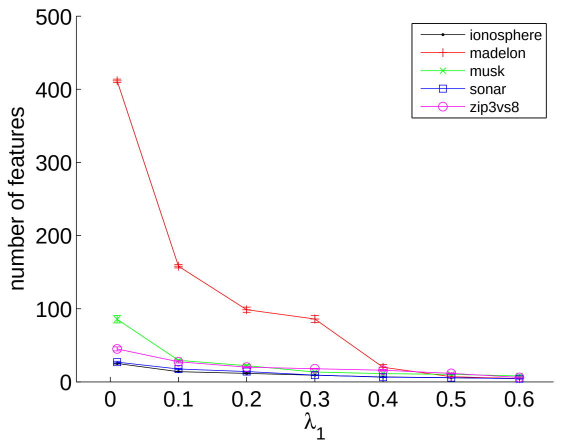
(a) The error rates of SMT



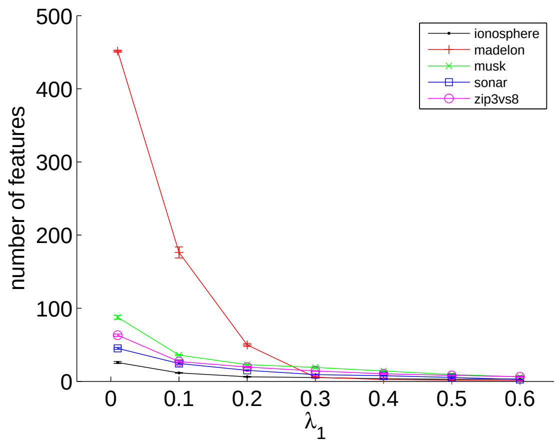
(b) The error rates of Lasso LR



(c) The depth of SMT

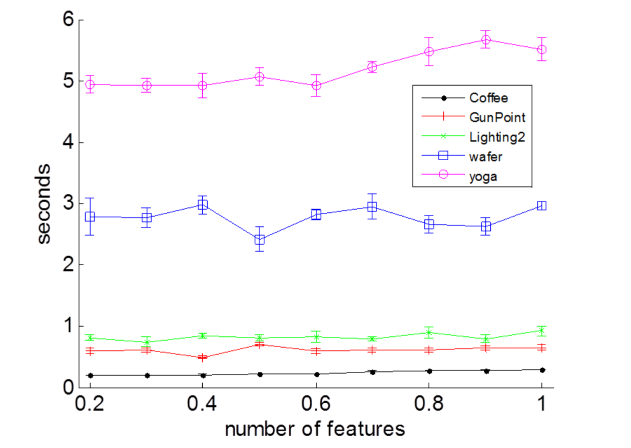


(d) The average number of features used in a node of SMT

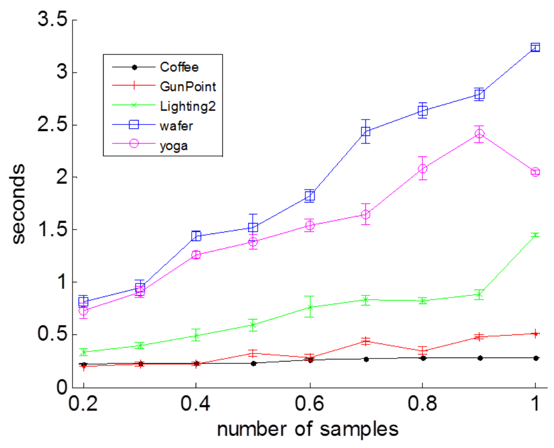


(e) The number of features used in Lasso LR

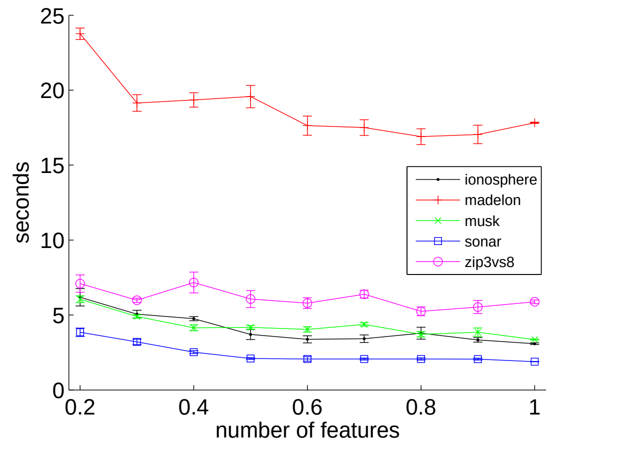
Figure 5: Sensitivity analysis of SMT and Lasso LR for non-time-series data. The above figures show the changes in the error rates, the number of features and the depth (only for SMT) due to the change of λ_1. Compared to Lasso LR, the error rates from SMT are much less sensitive to λ_1 for most data sets. As λ_1 increases, both the number of features used in Lasso LR and the average numbers of features used in a SMT non-leaf node decrease quickly. Also, the depth of SMT increases upto certain point and tends to stay around the same level with increasing λ_1.



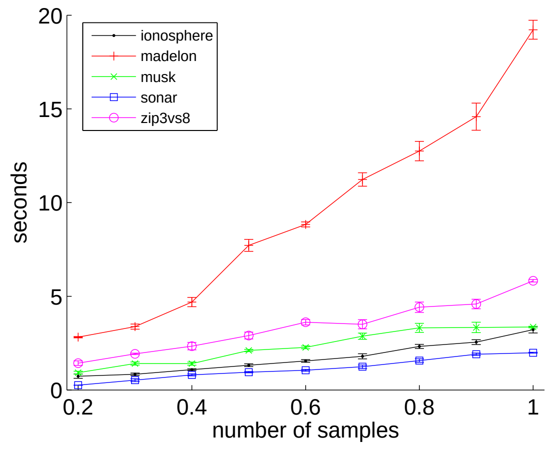
(a) SMT runtime versus the number of features for time series data



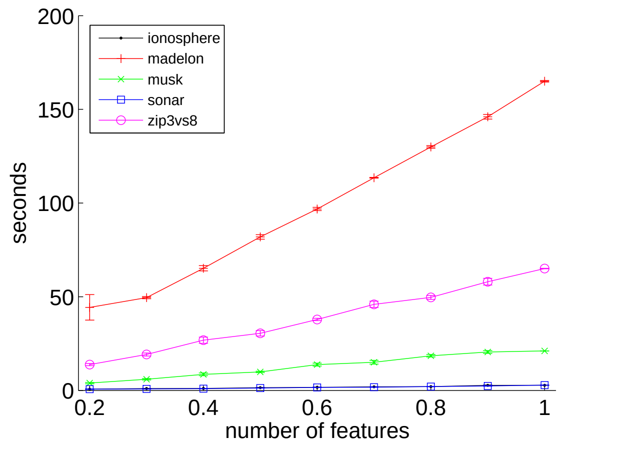
(b) SMT runtime versus the number of instances for time series data



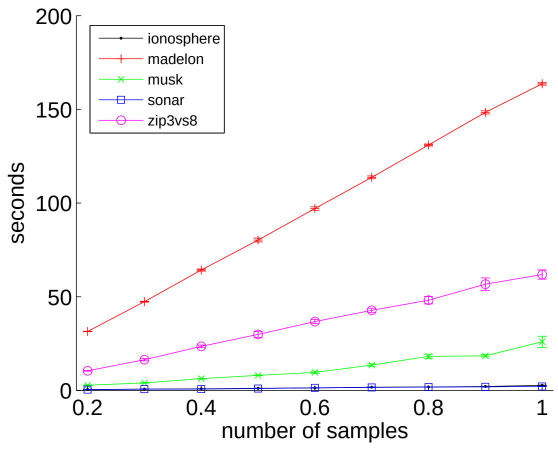
(c) SMT runtime versus the number of features for non-time-series data



(d) SMT runtime versus the number of instances for non-time-series data



(e) LMT runtime versus the number of features for non-time-series data



(f) LMT runtime versus the number of instances for non-time-series data

Figure 6: Runtime versus the number of features/instances (proportion of the original features/instances) for SMT and LMT. The features/instances were randomly chosen. The runtime of SMT is not strongly affected with the number of features over the range in these experiments. The LMT runtimes are much greater than SMT for the same number of features or instances.