Sparse Perceptron Decision Tree for Millions of Dimensions

Weiwei Liu and **Ivor W. Tsang**[*]
Centre for Quantum Computation and Intelligent Systems
University of Technology Sydney, Australia
liuweiwei863@gmail.com, ivor.tsang@uts.edu.au

Abstract

Due to the nonlinear but highly interpretable representations, decision tree (DT) models have significantly attracted a lot of attention of researchers. However, DT models usually suffer from the curse of dimensionality and achieve degenerated performance when there are many noisy features. To address these issues, this paper first presents a novel data-dependent generalization error bound for the perceptron decision tree (PDT), which provides the theoretical justification to learn a sparse linear hyperplane in each decision node and to prune the tree. Following our analysis, we introduce the notion of sparse perceptron decision node (SPDN) with a budget constraint on the weight coefficients, and propose a sparse perceptron decision tree (SPDT) algorithm to achieve nonlinear prediction performance. To avoid generating an unstable and complicated decision tree and improve the generalization of the SPDT, we present a pruning strategy by learning classifiers to minimize cross-validation errors on each SPDN. Extensive empirical studies verify that our SPDT is more resilient to noisy features and effectively generates a small, yet accurate decision tree. Compared with state-of-the-art DT methods and SVM, our SPDT achieves better generalization performance on ultrahigh dimensional problems with more than 1 million features.

Introduction

Due to the nonlinear, but highly interpretable representations (Murthy, Kasif, and Salzberg 1994), decision tree (DT) (Moret 1982; Safavian and Landgrebe 1991) has been a particularly powerful tool in various machine learning and data mining applications, ranging from Agriculture (McQueen et al. 1995), Astronomy (Salzberg et al. 1995) and Molecular Biology (Shimozono et al. 1994) to Financial analysis (Mezrich 1994).

Many early works focus on the DT, in which each node uses the value of a single feature (attribute) (Breiman et al. 1984; Quinlan 1993) for decision. Since the decision of those methods at each node is equivalent to an axis-parallel hyperplane in the feature space, they are called axis-parallel DT, which suffer from the curse of dimensionality. To improve the performance of axis-parallel DT, researchers developed perceptron decision tree (PDT) (Bennett et al. 2000) in which the test at a node uses the linear combinations of features. Clearly, PDT is the general form of axis-parallel DT.

Many real-world problems, like documents and Microarray data (Guyon and Elisseeff 2003), are represented as high dimensional vectors. Many features in high dimensional space are usually non-informative or noisy. Those noisy features will decrease the generalization performance of DT and derogate from their promising results for dealing with non-linear separable data, especially when they are many noisy features. For some applications like Microarray data analysis (Guyon and Elisseeff 2003), it is desired to identify a small set of features that represent the original feature space. Thus, it is imperative to develop a sparse DT with respect to input features and its corresponding theory.

Another drawback for DT is that its performance is sensitive to the tree structure, which is determined by the distribution of features. Thus, ensembles offer an effective technique for obtaining increased accuracy by combining the predictions of many different trees, like random forest feature selection (RFFS) (Hastie, Tibshirani, and Friedman 2001) and Gradient boosted feature selection (GBFS) (Xu et al. 2014). GBFS is an advanced gradient boosted DT (Friedman 2000), which employs gradient boosting to do feature selection in DT. GBFS, which belongs to axis-parallel DT, shows its state-of-the-art feature selection performance compared with l_1-regularized logistic regression (Lee et al. 2006), RFFS, and some other promising baselines. Cost-Sensitive Tree of Classifiers (CSTC) (Xu et al. 2013) is another state-of-the-art approach for feature selection in DT, which follows the PDT. Both are based on the l_1-regularized SVM (Zhu et al. 2003): minimizing l_1-regularization plus other loss functions. However, the theoretical and empirical analysis in (Tan, Tsang, and Wang 2014) have shown that l_1-regularized methods achieve suboptimal performance in feature selection, compared with the sparse model with an explicit budget constraint on the weights.

Our main contribution is first to present a novel generalization error bound for the PDT, which takes into account the distribution of the data, and provides the theoretical justification to learn a sparse hyperplane classifier in each decision node and prune the tree. From the analysis of our bound, the training error loss, the margin, the capacity of the kernel matrix defined on the training data (Tsang and Kwok 2006), and

[*]Corresponding author

the complexity of the tree are the dominant factors that affect the generalization performance of PDTs. In particular, our bound indicates that decreasing training error loss and the capacity of the kernel matrix, and enlarging the margin can yield better generalization performance. Suppose linear kernel is used, decreasing the capacity of the kernel matrix can be done via feature selection. Thereafter, we introduce the notion of sparse perceptron decision node (SPDN) to perform feature selection and optimize training error loss and margin simultaneously, and then develop a sparse perceptron decision tree (SPDT) to achieve nonlinear prediction performance. To obtain a stable DT structure, instead of conducting ensemble, we propose an objective to learn classifiers and to identify a universal feature subset that simultaneously minimizes the cross validation errors on each SPDN. After that, we further diminish the generalization error by pruning the tree. Finally, extensive empirical studies verify the effectiveness of our approach.

Related Work

OC1 (Murthy, Kasif, and Salzberg 1994) is a popular PDT, which belongs to a randomized algorithm that performs a randomized hill-climbing search to learn the classifiers and builds the tree in a top-down approach. Murthy, Kasif, and Salzberg (1994) show the competitive results of OC1 compared with C4.5 (Quinlan 1993) (which is the famous axis-parallel DT) and other baselines. Bennett et al. (2000) first provide the margin bound for the perceptron decision tree (PDT), then update the weights of OC1 with the maximum margin for each node. However, the results of (Bennett et al. 2000) show that the performance of modified-OC1 with the maximum margin for each node is similar with OC1. The main reason may be the data dependency of the bound comes through the margin only.

Another strategy is the combination of bagging and building ensembles of trees. For instance, Random Forests by Breiman (2001), is a popular method to control the generalization error of decision trees. While this paper focuses on single-tree performance on high-dimensional settings.

Region-specific and locally linear models (Jose et al. 2013; Oiwa and Fujimaki 2014) are an advanced technique for dealing with the non-linear problem. Local deep kernel learning (LDKL) (Jose et al. 2013) formulates the problem from a local kernel learning perspective where they learn tree-structured primal feature embedding and has shown its superior performance in (Jose et al. 2013). Partition-wise linear models (Oiwa and Fujimaki 2014) assign linear models to partitions and represent region-specific linear models by linear combinations of partition-specific models. They first formalize the problem as l_0 penalizing problem and then apply a convex relaxation with a combination of l_1 penalties. The model is similar with GBFS and CSTC, which is different in penalizing terms.

Theoretical Results

Following (Bartlett and Mendelson 2002; Shawe-Taylor and Cristianini 2004), this section provides the generalization error bound for the PDT, which takes into account the distribution of the data. We begin with some basic definitions. We denote the transpose of vector/matrix by the superscript $'$. Let $\odot$ represent the elementwise product sign. If $\mathcal{S}$ is a set, $|\mathcal{S}|$ denotes its cardinality. Let $\mathbb{R}$ represent real number, $\mathcal{X}$ be the input vector over $\mathbb{R}$ and $\mathcal{Y} = \{-1, 1\}$ be the output space. Suppose $sign(\cdot)$ represents sign function; $tr(\cdot)$ denotes the trace of matrix argument; $ln(\cdot)$ represents the natural logarithm and $\Theta(\cdot)$ returns 1 if its argument is greater than 0, and zero otherwise. $\hat{E_n}$ denotes the empirical risk. Let F be a class of functions mapping from $\mathcal{X}$ to $\mathbb{R}$. We follow the definitions in (Bartlett and Mendelson 2002) and use $R_n(F)$ and $G_n(F)$ to denote the Rademacher and Gaussian complexity of F, respectively.

Definition 1 (Generalized Decision Trees (GDT), Bennett et al. (2000)). *Given a space $\mathcal{X}$ and a set of Boolean functions $F_{GDT} = \{f_{GDT} : \mathcal{X} \to \{0, 1\}\}$, the class of Generalized Decision Trees over F_{GDT} is the set of functions that can be implemented using a binary tree where each internal node is labeled with an element of F_{GDT}, and each leaf is labeled with either 1 or 0.*

Definition 2 (Perceptron Decision Tree (PDT), Bennett et al. (2000)). *Given $\mathcal{X} = \mathbb{R}^m$, and assume g is a linear real function over $\mathcal{X}$ and $w \in \mathbb{R}^m$, a Perceptron Decision Tree (PDT) is a GDT over*

$$F_{PDT} = \{h_w : h_w(x) = \Theta(g_w(x)), g_w(x) = w'x + b\}$$

Next, we introduce two important Theorems.

Theorem 1 (Bartlett and Mendelson (2002)). *Let D be a probability distribution on $\mathcal{X} \times \mathcal{Y}$ and let F be a set of real valued functions defined on $\mathcal{X}$, with $\sup\{|\mathfrak{F}(x)| : \mathfrak{F} \in F\}$ finite for all $x \in \mathcal{X}$. Suppose that $\phi : \mathbb{R} \to [0, 1]$ satisfies $\phi(\alpha) \geq 1$ ($\alpha \leq 0$) and is Lipschitz with constant L, then with probability at least $1 - \delta$ with respect to training samples $S = \{(x_1, y_1), \cdots, (x_n, y_n)\}$ drawn independently according to D, every function in F satisfies*

$$\begin{aligned} &P_D(y \neq sign(\mathfrak{F}(x))) \\ &\quad \leq \hat{E_n}(\phi(y\mathfrak{F}(x))) + 2LR_n(F) + \sqrt{\frac{ln(2/\delta)}{2n}} \end{aligned}$$

Theorem 2 (Shawe-Taylor and Cristianini (2004)). *Fix γ and let $\mathcal{F}$ be the class of functions mapping from $\mathcal{X} \times \mathcal{Y}$ to $\mathbb{R}$ given by $f(x, y) = -yg(x)$ ($x \in \mathcal{X}$, $y \in \mathcal{Y}$), where g is a linear function in a feature space induced by a kernel $k(\cdot, \cdot)$ with Euclidean norm at most 1. Let $S = \{(x_1, y_1), \cdots, (x_n, y_n)\}$ be drawn independently according to a probability distribution D and fix $\delta \in (0, 1)$. Then, with probability at least $1 - \delta$ over samples of size n we have*

$$\begin{aligned} P_D(y \neq sign(g(x))) &= \mathbb{E}_D[\Theta(-yg(x))] \\ &\leq \frac{1}{n\gamma}\sum_{i=1}^{n}\xi_i + \frac{4}{n\gamma}\sqrt{tr(K)} + 3\sqrt{\frac{ln(2/\delta)}{2n}} \end{aligned}$$

where K is a kernel matrix defined on the training data with $k(\cdot, \cdot)$ and $\xi_i = \max(0, \gamma - y_i g(x_i))$.

The generalization error bound for PDT is given as follows:

Theorem 3. *Let T be a PDT, where the linear functions g_w in T are with the Euclidean norm at most 1. Suppose that the depth of T is no more than d, and it contains no more than M leaves. Let $S = \{(x_1, y_1), \cdots, (x_n, y_n)\}$ be the n sample, which is generated independently according to a probability distribution D. Then, we can bound the generalization error with probability greater than $1 - \delta$ to be less than*

$$\sum_l \sum_{i=1}^{d_l} \mathcal{D}_i^l + \frac{5cdMG_n(T)}{2} + \big((3d+1)\sqrt{n}M + 1\big)\sqrt{\frac{ln(6/\delta)}{2n}}$$

where c is a constant, $d_l (d_l \leq d)$ is the depth of leaf l, $\mathcal{D}_i^l = \frac{1}{n_l\gamma_i^l}\sum_{j=1}^{n_l}\xi_{i,j}^l + \frac{4}{n_l\gamma_i^l}\sqrt{tr(K_l)}$, n_l denotes the number of samples $S_l = \{(x_1^l, y_1^l), \cdots, (x_{n_l}^l, y_{n_l}^l)\}$ reaching leaf l, K_l is the kernel matrix for $\{x_1^l, \cdots, x_{n_l}^l\}$, $\xi_{i,j}^l = \max\big(0, \gamma_i^l - y_{i,j}^l g_i^l(x_j^l)\big)$, where $y_{i,j}^l$ represents the correct output for input x_j^l in decision node i and γ_i^l denotes the corresponding margin.

Proof. For a tree of depth d, the indicator function of a leaf is a conjunction of no more than d decision functions. More specifically, if the decision tree consists of decision nodes chosen from a class T of Boolean functions, the indicator function of leaf l (which takes value 1 at a point x if x reaches l and 0 otherwise) is a conjunction of d_l functions from T, where d_l is the depth of leaf l. We can represent the function computed by the tree as follows:

$$f(x) = \sum_l \sigma_l \bigwedge_{i=1}^{d_l} h_{l,i}(x)$$

where the sum is over all leaves l, $\sigma_l \in \{\pm 1\}$ is the label of leaf l, $h_{l,i} \in T$, and the conjunction is understood to map to $\{0, 1\}$. Let F be this class of functions. Choose a family $\{\phi_L : L \in \mathbb{N}\}$ of cost functions such that each ϕ_L dominates the step function $\Theta(-yf(x))$ and has a Lipschitz constant L. For each L, Theorem 1 implies that with probability at least $1 - \delta_1$:

$$\begin{aligned} &P(y \neq f(x)) \\ &\leq \hat{E}_n(\phi_L(yf(x))) + 2LR_n(F) + \sqrt{\frac{ln(2/\delta_1)}{2n}} \end{aligned}$$

Define $\phi_L(\alpha)$ to be 1 if $\alpha \leq 0$, $1 - L\alpha$ if $0 < \alpha \leq 1/L$, and 0 otherwise. Let $\tilde{P}_n(l)$ denotes the proportion of all the samples in S which reach leaf l and are correctly classified. Assume $S_l = \{(x_1^l, y_1^l), \cdots, (x_{n_l}^l, y_{n_l}^l)\}$ reach leaf l with $|S_l| = n_l$. Then we have

$$\begin{aligned} &\hat{E}_n(\phi_L(yf(x))) \\ &= \sum_l \frac{\sum_{(x^l,y^l)\in S_l}\Theta(-y^l f(x^l))}{n} + \sum_l \tilde{P}_n(l)\phi_L(1) \\ &= \sum_l \frac{\sum_{(x^l,y^l)\in S_l}\Theta(-y^l f(x^l))}{n} + \sum_l \tilde{P}_n(l) max(0, 1-L) \end{aligned} \tag{1}$$

If no sample reaches leaf l, the sum term of leaf l in the Eq.(1) is zero. Here we consider $1 \leq n_l < n$. As $L \in \mathbb{N}$, the second term in Eq.(1) is zero. Following the generalization error bounds w.r.t. Rademacher complexity in (Bartlett and Mendelson 2002), with probability at least $1 - \delta_2$, we get:

$$\begin{aligned} &\frac{\sum_{(x^l,y^l)\in S_l}\Theta(-y^l f(x^l))}{n} \leq \frac{\sum_{(x^l,y^l)\in S_l}\Theta(-y^l f(x^l))}{n_l} \\ &\leq P(y^l \neq f(x^l)) + \frac{R_n(F)}{2} + \sqrt{\frac{ln(2/\delta_2)}{2n_l}} \end{aligned} \tag{2}$$

The probability of x^l misclassified is equal to the probability of at least one decision node from the root to a leaf l making the errors. Suppose $\{g_1^l, \cdots, g_{d_l}^l\}$ are the linear functions with the Euclidean norm at most 1 and associated with each decision node from the root to a leaf l. The corresponding margins are $\{\gamma_1^l, \cdots, \gamma_{d_l}^l\}$. For input $x^l \in \{x_1^l, \cdots, x_{n_l}^l\}$, the correct output of each decision node from the root to a leaf l is denoted by $\{y_1^l, \cdots, y_{d_l}^l\}$, where $y_i^l \in \{y_{i,1}^l, \cdots, y_{i,n_l}^l\}$. By the union bound and Theorem 2, with probability at least $1 - \delta_3$, we get the following:

$$\begin{aligned} &P(y^l \neq f(x^l)) \\ &= P\Big(\bigcup_{i=1}^{d_l}(y_i^l \neq sign(g_i^l(x^l)))\Big) \\ &\leq \sum_{i=1}^{d_l} P(y_i^l \neq sign(g_i^l(x^l))) \\ &\leq \sum_{i=1}^{d_l}\Big(\frac{1}{n_l\gamma_i^l}\sum_{j=1}^{n_l}\xi_{i,j}^l + \frac{4}{n_l\gamma_i^l}\sqrt{tr(K_l)}\Big) + 3d_l\sqrt{\frac{ln(2/\delta_3)}{2n_l}} \end{aligned} \tag{3}$$

Let $\delta_1 = \delta_2 = \delta_3 = \delta/3$ and $L = M$. The results of Eq.(1), Eq.(2) and Eq.(3) imply that with probability, at least $1 - \delta$

$$\begin{aligned} &P(y \neq f(x)) \\ &\leq \sum_l \sum_{i=1}^{d_l} \mathcal{D}_i^l + \frac{5MR_n(F)}{2} + \big((3d+1)\sqrt{n}M+1\big)\sqrt{\frac{ln(6/\delta)}{2n}} \end{aligned}$$

where $\mathcal{D}_i^l = \frac{1}{n_l\gamma_i^l}\sum_{j=1}^{n_l}\xi_{i,j}^l + \frac{4}{n_l\gamma_i^l}\sqrt{tr(K_l)}$. Let c is a constant. Theorem 12, Theorem 16 and Lemma 4 in (Bartlett and Mendelson 2002) imply that $R_n(F) \leq cdG_n(T)$, which implies the result. □

Theorem 3 reveals important factors to reduce the generalization error bound for the PDT model: the first is to decrease the training error loss and the capacity of the kernel matrix for each node, and the second is to enlarge the margin for each node. We first enforce an upper bound budget $\mathfrak{B}$ for the capacity of the kernel matrix, *i.e.* $tr(K_l) \leq \mathfrak{B}$. Suppose there are n samples $S = \{(x_1, y_1), \cdots, (x_n, y_n)\}$, $x_i \in \mathbb{R}^m$ ($x_i = [x_{i,1}, \cdots, x_{i,m}]'$). Since we use linear kernel here, decreasing the capacity of the kernel matrix can be transformed to a budget constraint on the weight coefficients, which leads to a feature selection problem for the linear decision function. Mathematically, let $\hat{K}$ denote the kernel matrix on the re-scaled instance: $\hat{x}_i = x_i \odot \varrho$, where $\varrho = [\varrho_1, \cdots, \varrho_m]' \in \{0,1\}^m$ represents a feature selection vector. The capacity constraint on the kernel matrix is $tr(\hat{K}) \leq \mathfrak{B}$, then $tr(\hat{K}) = \sum_{i=1}^n \varrho'(x_i \odot x_i) \leq n\tilde{x}_{max} \sum_{j=1}^m \varrho_j$, where $\tilde{x}_{max}$ be the maximum element in $[x_1 \odot x_1, \cdots, x_n \odot x_n]$. Let $\mathfrak{B}/n\tilde{x}_{max} = B$, where B is a budget constraint on the weight coefficients. Thus $tr(\hat{K}) \leq \mathfrak{B}$ can be transformed to $\sum_{j=1}^m \varrho_j \leq B, \varrho \in \{0,1\}^m$. Then we learn a sparse linear decision function on each node by minimizing the training error loss and maximizing the margin at the same time. Based on Theorem 3, we hereinafter introduce the notion of sparse perceptron decision node (SPDN).

Definition 3 (Sparse Perceptron Decision Node (SPDN))**.** *Assume there are n samples $S = \{(x_1, y_1), \cdots, (x_n, y_n)\}$ drawn independently according to a probability distribution D, where $x_i \in \mathbb{R}^m$ ($x_i = [x_{i,1}, \cdots, x_{i,m}]'$) and $y_i \in \{\pm 1\}$. $\varrho = [\varrho_1, \cdots, \varrho_m]' \in \{0,1\}^m$ represents a feature selection vector. Let $\hat{K}$ denote the kernel matrix on the re-scaled instance: $\hat{x}_i = x_i \odot \varrho$ and the capacity constraint on the kernel matrix can be transformed as $\sum_{j=1}^m \varrho_j \leq B$ with a budget B. A sparse perceptron decision node contains a linear decision function $g_w(x) = w'(x_i \odot \varrho + b), ||w||^2 = 1$, where the parameters are obtained by solving the following problem :*

$$\begin{aligned} &\min_{\varrho \in \mathcal{E}} \min_{w,\xi,\gamma} -\gamma + C\sum_{i=1}^n \xi_i \\ &s.t. \quad y_i w'(x_i \odot \varrho + b) \geq \gamma - \xi_i, \xi_i \geq 0, i = 1, \cdots, n \\ &\qquad ||w||^2 = 1, \gamma \geq 0 \end{aligned} \tag{4}$$

where $\mathcal{E} = \{\varrho \in \{0,1\}^m | \sum_{j=1}^m \varrho_j \leq B\}$ and B is a budget constraint on the weight coefficients.

Sparse Perceptron Decision Tree

In this section, we first present an algorithm for SPDT, then develop the technique for learning a classifier at SPDN. After that, we present a pruning strategy to improve the generalization of SPDT.

Algorithm for SPDT

We build an SPDT in a top-down approach. Starting from the root, we train one linear decision function on each SPDN and use it to split the training data into two child nodes. This process continues until the remaining data at the node cannot be further split by the induced classifier. The details are stated in Algorithm 1.

Decision Function Learning at SPDN

For the sake of clarity, we consider the function without an offset, although the model can be extended to the function with the offset. For the inner minimization problem of Eq.(4), the corresponding Lagrangian function is: $\mathcal{L}(w, \xi, \gamma, \alpha, \lambda, \psi, \nu) = -\gamma + C\sum_{i=1}^n \xi_i - \sum_{i=1}^n \alpha_i(y_i w'(x_i \odot \sqrt{\varrho}) - \gamma + \xi_i) + \lambda(\|w\|^2 - 1) - \sum_{i=1}^n \psi_i \xi_i - \nu\gamma$, with $\alpha_i, \lambda, \psi_i, \nu \geq 0$. Differentiating w.r.t.

Algorithm 1 Sparse Perceptron Decision Tree (SPDT)

```
Input: Training data $S = \{(x_1, y_1), \cdots, (x_n, y_n)\}$, where
    $x_i \in \mathbb{R}^m$ and $y_i \in \{\pm 1\}$.
Output: A sparse perceptron decision tree
1: while SPDN contains more than one class do
2:     train one classifier for this SPDN
3:     split the training data of this node into left child if
       those instances are predicted to be positive by the
       classifier and call Algorithm 1 with split data as the
       input.
4:     split the rest of training data of this node into right
       child and call Algorithm 1 with the remaining data as
       the input.
5: end while
6: Return the complete sparse perceptron decision tree.
```

the primal variables and substituting the relations obtained into the primal, we obtain the following dual objective function: $\mathcal{L}(\alpha, \lambda, \varrho) = -\frac{1}{4\lambda}\left\| \sum_{i=1}^n \alpha_i y_i (x_i \odot \varrho)\right\|^2 - \lambda$. Optimising λ gives $\lambda^* = \frac{1}{2}\left\| \sum_{i=1}^n \alpha_i y_i (x_i \odot \varrho)\right\|$, resulting in $\mathcal{L}(\alpha, \varrho) = -\left\| \sum_{i=1}^n \alpha_i y_i (x_i \odot \varrho)\right\|$. Thus, Eq.(4) can be equivalently reformulated as the following problem:

$$\min_{w,\varrho} SPDN(w, \varrho, S) = \min_{\varrho \in \mathcal{E}} \max_{\alpha \in \mathcal{A}} -\frac{1}{2}\left\| \sum_{i=1}^n \alpha_i y_i (x_i \odot \varrho)\right\|^2 \tag{5}$$

where $\mathcal{A} = \{\alpha | \sum_{i=1}^n \alpha_i \geq 1, 0 \leq \alpha_i \leq C, i = 1, \cdots, n\}$.

Though Eq.(5) is a mixed-integer problem, we can adapt the feature generating machine (FGM) (Tan, Wang, and Tsang 2010; Zhai et al. 2012; Tan, Tsang, and Wang 2013; 2014) to solve it efficiently. As shown in Algorithm 1 of (Tan, Tsang, and Wang 2014), the optimization strategy of FGM is the same as the cutting plane algorithm, which involves two major steps: **worst-case analysis** and **master-problem optimization**.

Worst-case analysis For the fixed α, we need to solve

$$\max_{\varrho} \frac{1}{2}\left\| \sum_{i=1}^n \alpha_i y_i (x_i \odot \varrho)\right\|^2 : \sum_{j=1}^m \varrho_j \leq B, \varrho \in \{0,1\}^m \tag{6}$$

This is transformed as $\max_\varrho \sum_{j=1}^m \frac{1}{2}(\sum_{i=1}^n \alpha_i y_i x_{i,j})^2 \varrho_j$ subjected to $\sum_{j=1}^m \varrho_j \leq B, \varrho \in \{0,1\}^m$, which is a linear integer programming (LIP) problem. We speed up solving this problem by the sorting method.

Master-problem optimization The solution ϱ_t of Eq.(6) at each worst-case analysis is added to a candidate set, then we employ a multiple kernel learning (MKL) model with base kernels corresponding to each ϱ_t in the candidate set to solve the convex relaxation of Eq.(5). The resultant MKL problem can be solved efficiently by the modified accelerating proximal gradient (APG) method developed in (Tan, Tsang, and Wang 2014), which takes $\mathcal{O}(nTB)$ time complexity, where T is the number of the worst-case analyses.

SPDT Pruning

Theorem 3 reveals that the generalized bound also depends on the depth and leaves of the tree, which appear in the sec-

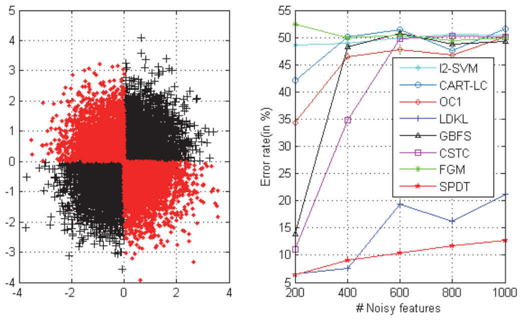

Figure 1: Testing error rates (in %) on the XOR synthetic data with different # noisy features.

ond and third terms of the bound. Inspired by this, we can reduce the depth and leaves of the tree by pruning the tree to further diminish the generalization error. We follow the pruning strategy in (Xu et al. 2013) to conduct pruning for SPDT. To this end, we present to use the cross validation (CV) for each node to estimate the generalization error. If the sum of CV errors of child nodes is bigger than that of the parent node, we remove those child nodes.

It is obvious that any slight changes in the data distribution result in generating different feature subsets for many feature selection methods, like (Xu et al. 2013; 2014). To identify a stable and unique feature subset across different folds of CV, we introduce the universal feature selection vector ϱ^{uni}, which is selected by the following formulation:

$$\min_{w^v, \varrho^{uni}} \sum_v SPDN(w^v, \varrho^{uni}, S^v) \tag{7}$$

where S^v represents the $v-th$ fold of training data, w^v is the corresponding weight vector. To solve Eq.(7), we first conduct the worst-case analysis using the sum of objectives of each data fold. Then, we get the universal feature selection vector ϱ^{uni} and based on this, we adapt the efficient Master-problem solvers in (Tan, Tsang, and Wang 2014) to get the w^v for each data fold separately. We repeat those procedures until convergence. This framework is called embedded cross validation. By using this method, we can find an intrinsic set of features.

For prediction, we aggregate the outputs from classifiers, which is generated from each data fold, by means of voting.

Experiment

In this section, we conduct comprehensive experiments to evaluate the proposed method on both synthetic and high dimensional real-world data sets. Most data sets are collected from this website[1]. pcmac data set is from (Xu et al. 2014). Data sets used in this paper is described in Table 1.

We compare SPDT with a number of baseline methods: standard SVM (l_2-SVM)[2], CART-LC (Breiman et al. 1984), OC1 (Murthy, Kasif, and Salzberg 1994), LDKL (Jose et al. 2013), GBFS (Xu et al. 2014), CSTC (Xu et al. 2013) and FGM (Tan, Tsang, and Wang 2014). We use the linear classification/regression package LIBLINEAR (Fan et al. 2008) with L2-regularized square hinge loss (primal) to implement standard SVM. We compare with LDKL, which uses composite non-linear kernels, achieves significantly better classification accuracies as compared to localized multiple kernel learning (Gönen and Alpaydin 2008) (LMKL, which is competitive with RBF-SVM) and other state-of-the-art nonlinear methods. Therefore, we do not compare with other nonlinear SVMs. The codes of other baseline methods are provided by their authors.

We use 5-fold cross validation to prune SPDT. Following the parameter settings in (Tan, Tsang, and Wang 2014), B is chosen in a range of $\{2, 5, 10, 20, 50, 100, 150, 200, 250\}$ for the rcv data set and $\{0.01m, 0.02m, \cdots, 0.09m\}$ for other data sets; C is selected using 5-fold cross validation over the range $\{0.001, 0.01, 0.1, 5, 10\}$ for the first three data sets and we fix $C = 5$ for larger data sets like epsilon and rcv. The tree-depth is fixed to 3 in LDKL, following the settings in (Oiwa and Fujimaki 2014).

Table 1: Datasets used in the experiments

DATASET	# FEATURES	# TRAINING	# TESTING
colon	2,000	32	30
pcmac	3,289	1,555	388
gisette	5,000	6,000	1,000
epsilon	2,000	21,767	14,511
rcv	47,236	12,146	8,096
news20	1,355,191	3,000	2,000

Experiments on Synthetic Data

We compare the performance of different methods on the XOR synthetic data with different numbers of noisy features, which is non-linearly separable. Following the strategy in (Tan, Tsang, and Wang 2014), we generate the synthetic data as follows: At first, we generate 10,000 training instances, 2,000 testing instances and 200 features. The first two dimensions are informative features, which are sampled from the i.i.d. Gaussian distribution $\mathcal{N}(0, 1)$. The remaining dimension are noisy features, which are sampled from the i.i.d. Uniform distribution $\mathcal{U}(0, 1)$. The output y is decided by those two informative features. For instance, if the first two features have the same sign, then y is 1 and -1 otherwise. Then, we gradually increase the noisy features to the dataset and test whether the considered methods can successfully identify the former two informative features and have stable and accurate classification performance. Figure 1 (left panel) shows two informative features. We fix $B = 1$ for SPDT.

Figure 1 (right panel) shows that: 1) l_2-SVM cannot work on the XOR data set. 2) FGM has shown impressive results in (Tan, Tsang, and Wang 2014), but it cannot work on the XOR data set as well. 3) When the number of noisy features equals 200, CART-LC and OC1 are a bit better than l_2-SVM; GBFS and CSTC are much better than other baselines. However, CART-LC, OC1, GBFS and CSTC are sensitive to the

[1]http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/
[2]https://www.csie.ntu.edu.tw/∼cjlin/liblinear/

Table 2: Testing error rates (in %) for different methods. Numbers in parentheses indicate the depth of tree. The best results are in bold.

DATASET	l_2-SVM	CART-LC	OC1	LDKL	SPDT
colon	20.00	40.91(2)	33.33(2)	16.67	**13.33**(1)
pcmac	7.47	16.22(2)	9.68(6)	7.99	**7.22**(2)
gisette	2.30	48.17(5)	18.83(6)	2.80	**2.20**(2)
epsilon	13.24	49.40(2)	47.66(4)	15.36	**12.45**(8)
rcv	4.05	17.13(6)	48.68(3)	3.90	**3.85**(5)

number of noisy features. 4) LDKL and SPDT are most successful, while our model generates more stable and accurate results.

Experiments on Real-world Data

Classification Performance To verify the effectiveness of conducting feature selection in the decision tree, we evaluate the classification performance of SPDT on all data sets compared with l_2-SVM, CART-LC, OC1 and LDKL. The results are shown in Table 2.

The results of Table 2 show that: 1) CART-LC and OC1 generally underperform on all data sets. The results support the argument of this paper: DT models usually suffer from the curse of dimensionality and achieve degenerated performance on high dimensional data sets, which boosts our approach. 2) SPDT gets a lower error rate than l_2-SVM and LDKL. It is important to generate different hyperplanes with maximum-margin based on different sets of features. Thus, the results verify the effectiveness of conducting feature selection in the DT. 3) SPDT has the best performance compared with baselines, while generating a small tree.

Feature Selection Performance We evaluate the feature selection performance of SPDT on pcmac, gisette, epsilon and rcv data sets compared with sparse classifier models, such as GBFS, CSTC and FGM. CSTC runs out of memory on rcv. The results of Figure 2 show that: 1) FGM and SPDT clearly outperform GBFS and CSTC in terms of feature selection performance, which verifies that the sparse models with an explicit budget constraint on the weights are more effective than l_1-SVM based approaches. 2) SPDT outperforms FGM. Thus, it is imperative to utilize the tree-structure to conduct feature selection.

High-dimensional Results We evaluate our method on the news20 data set with 1,355,191 features. As CART-LC, OC1, LDKL, GBFS and CSTC are very slow on this data set, we compare our method with l_2-SVM and FGM only on this data set. For l_2-SVM, the error rate is 13.2%. Other results are reported in Table 3. For news20, when selecting 1,000 features in high dimensional results, the training time for our SPDT is 31.8 seconds. So SPDT can efficiently handle the high-dimensional data set with 10^6 features and achieve superior performance compared with FGM and SVMs.

Training Time Some baselines like l_2-SVM, GBFS and LDKL are written in C++; while CSTC and SPDT are implemented in Matlab. In each SPDN, the learning model can be solved by efficient feature selection methods, such as FGM[3]. We use the rcv data set for time comparison. The results are shown in Table 4. Since CART-LC, OC1 and CSTC are very slow on this data set, we cannot report their training time here. Even though GBFS and LDKL are written in C++, from Table 4, we can see that, the training of our method is much faster than that of GBFS, and it is also much faster than LDKL, which takes 8019.3s.

Table 3: Testing error rates (in %) for FGM and our method on news20 data set. The better results are in bold.

# Selected features	FGM	SPDT
300	13.85	**3.40**
800	9.15	**2.70**
1,000	9.40	**2.45**

Table 4: Training time (in second) for GBFS and our method on rcv data set. The faster results are in bold.

# Selected features	GBFS	SPDT
100	874.2s	**5.9**s
500	5214.8s	**12.0**s
900	8523.3s	**41.9**s

Conclusion

In this paper, we first develop a new generalization error bound for the PDT, where the data dependency of the bound comes through the training error loss, the margin and the capacity of the kernel matrix defined on the training data. It provides the theoretical justification to learn a sparse linear hyperplane in each decision node and to prune the SPDT. Our analysis reveals a new insight for the design of decision tree algorithms. Compared with state-of-the-art baselines, the results on the synthetic data set show that SPDT is more resilient to noisy features; and empirical studies on real-world data sets demonstrate that SPDT generates a small, yet more accurate decision tree for high dimensional data, and can identify more informative features than state-of-the-art feature selection methods. Moreover, the training of our SPDT is much more efficient than other decision tree algorithms. In future, we will explore how to extend the work to multi-class problems.

Acknowledgments

This project is supported by the ARC Future Fellowship FT130100746 and ARC grant LP150100671.

References

Bartlett, P. L., and Mendelson, S. 2002. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research* 3:463–482.

[3]We modified the FGM software that is available at http://www.tanmingkui.com/fgm.html.

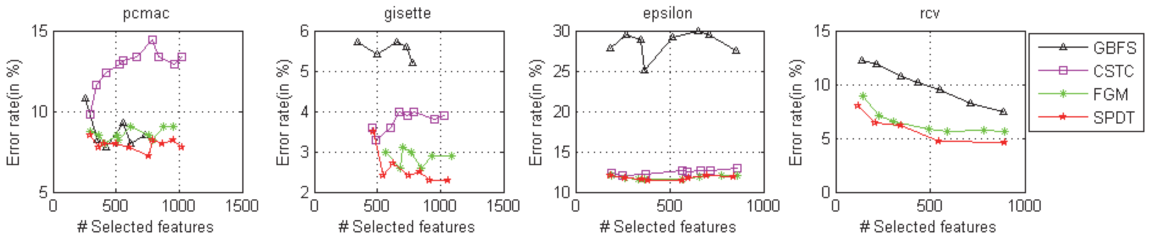

Figure 2: Testing error rates (in %) vs. # selected features for different methods on pcmac, gisette, epsilon and rcv data sets.

Bennett, K. P.; Cristianini, N.; Shawe-Taylor, J.; and Wu, D. 2000. Enlarging the margins in perceptron decision trees. *Machine Learning* 41(3):295–313.

Breiman, L.; Friedman, J. H.; Olshen, R. A.; and Stone, C. J. 1984. *Classification and Regression Trees*. CA: Wadsworth International Group.

Breiman, L. 2001. Random forests. *Machine Learning* 45:5–32.

Fan, R.-E.; Chang, K.-W.; Hsieh, C.-J.; Wang, X.-R.; and Chih-Jen, L. 2008. Liblinear: A library for large linear classification. *Journal of Machine Learning Research* 9:1871–1874.

Friedman, J. H. 2000. Greedy function approximation: A gradient boosting machine. *Annals of Statistics* 29:1189–1232.

Gönen, M., and Alpaydin, E. 2008. Localized multiple kernel learning. In *Proceedings of the 25th International Conference on Machine Learning*, 352–359.

Guyon, I., and Elisseeff, A. 2003. An introduction to variable and feature selection. *Journal of Machine Learning Research* 3:1157–1182.

Hastie, T.; Tibshirani, R.; and Friedman, J. 2001. *The Elements of Statistical Learning*. New York: Springer New York Inc.

Jose, C.; Goyal, P.; Aggrwal, P.; and Varma, M. 2013. Local deep kernel learning for efficient non-linear svm prediction. In *Proceedings of the 30th International Conference on Machine Learning*, 486–494.

Lee, S.-I.; Lee, H.; Abbeel, P.; and Ng, A. Y. 2006. Efficient l1 regularized logistic regression. In *The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference*, 401–403.

McQueen, R. J.; S.R., G.; C.G., N.-M.; and I.H., W. 1995. Applying machine learning to agricultural data. *Computers and Electronics in Agriculture* 12(4):275–293.

Mezrich, J. J. 1994. When is a tree a hedge? *Financial Analysts Journal* 75–81.

Moret, B. M. E. 1982. Decision trees and diagrams. *Computing Surveys* 14(4):593–623.

Murthy, S. K.; Kasif, S.; and Salzberg, S. 1994. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research* 2:1–32.

Oiwa, H., and Fujimaki, R. 2014. Partition-wise linear models. In *Advances in Neural Information Processing Systems 27*, 3527–3535.

Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning*. USA: Morgan Kaufmann Publishers Inc.

Safavian, S. R., and Landgrebe, D. A. 1991. A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man and Cybernetics* 21(3):660–674.

Salzberg, S.; Chandar, R.; Ford, H.; Murthy, S. K.; and White, R. L. 1995. Decision trees for automated identification of cosmic-ray hits in hubble space telescope images. *Publications of the Astronomical Society of the Pacific* 107:1–10.

Shawe-Taylor, J., and Cristianini, N. 2004. *Kernel Methods for Pattern Analysis*. New York: Cambridge University Press.

Shimozono, S.; Shinohara, A.; Shinohara, T.; Miyano, S.; Kuhara, S.; and Arikawa, S. 1994. Knowledge acquisition from amino acid sequences by machine learning system bonsai. *Transactions of the Information Processing Society of Japan* 35(10):2009–2018.

Tan, M.; Tsang, I. W.; and Wang, L. 2013. Minimax sparse logistic regression for very high-dimensional feature selection. *IEEE Transactions on Neural Networks and Learning Systems* 24(10):1609–1622.

Tan, M.; Tsang, I. W.; and Wang, L. 2014. Towards ultrahigh dimensional feature selection for big data. *Journal of Machine Learning Research* 15(1):1371–1429.

Tan, M.; Wang, L.; and Tsang, I. W. 2010. Learning sparse svm for feature selection on very high dimensional datasets. In *Proceedings of the 27th International Conference on Machine Learning*, 1047–1054.

Tsang, I. W., and Kwok, J. 2006. Efficient hyperkernel learning using second-order cone programming. *IEEE Transactions on Neural Networks* 17(1):48–58.

Xu, Z. E.; Kusner, M. J.; Weinberger, K. Q.; and Chen, M. 2013. Cost-sensitive tree of classifiers. In *Proceedings of the 30th International Conference on Machine Learning*, 133–141.

Xu, Z. E.; Huang, G.; Weinberger, K. Q.; and Zheng, A. X. 2014. Gradient boosted feature selection. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 522–531.

Zhai, Y.; Tan, M.; Tsang, I. W.; and Ong, Y.-S. 2012. Discovering support and affiliated features from very high dimensions. In *Proceedings of the 29th International Conference on Machine Learning*.

Zhu, J.; Rosset, S.; Hastie, T.; and Tibshirani, R. 2003. 1-norm support vector machines. In *Advances in Neural Information Processing Systems 16*, 49–56.