Journal of Machine Learning Research 18 (2017) 1-36 Submitted 9/16; Revised 3/17; Published 7/17

Making Decision Trees Feasible in Ultrahigh Feature and Label Dimensions

Weiwei Liu LIUWEIWEI863@GMAIL.COM
Centre for Artificial Intelligence
FEIT, University of Technology Sydney
NSW 2007, Australia
and
School of Computer Science and Engineering
The University of New South Wales, Sydney
NSW 2052, Australia

Ivor W. Tsang IVOR.TSANG@UTS.EDU.AU
Centre for Artificial Intelligence
FEIT, University of Technology Sydney
NSW 2007, Australia

Editor: Inderjit Dhillon

Abstract

Due to the non-linear but highly interpretable representations, decision tree (DT) models have significantly attracted a lot of attention of researchers. However, it is difficult to understand and interpret DT models in ultrahigh dimensions and 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 budget-aware classifier (BAC) with a budget constraint on the weight coefficients, and propose a supervised budgeted tree (SBT) algorithm to achieve non-linear prediction performance. To avoid generating an unstable and complicated decision tree and improve the generalization of the SBT, we present a pruning strategy by learning classifiers to minimize cross-validation errors on each BAC. To deal with ultrahigh label dimensions, based on three important phenomena of real-world data sets from a variety of application domains, we develop a sparse coding tree framework for multi-label annotation problems and provide the theoretical analysis. Extensive empirical studies verify that 1) SBT is easy to understand and interpret in ultrahigh dimensions and is more resilient to noisy features. 2) Compared with state-of-the-art algorithms, our proposed sparse coding tree framework is more efficient, yet accurate in ultrahigh label and feature dimensions.

Keywords: Classification, Ultrahigh Feature Dimensions, Ultrahigh Label Dimensions, Perceptron Decision Tree

1. Introduction

Decision tree (DT) (Moret, 1982; Safavian and Landgrebe, 1991) has been identified as one of the top 10 data mining and machine learning algorithms (Wu et al., 2008). Due to the non-linear, but highly interpretable representations (Murthy et al., 1994), DT has played a vital role in various data mining and machine learning applications, ranging from Agriculture (McQueen et al., 1995), Astronomy (Salzberg et al., 1995), Medicine (Czajkowski et al., 2014) 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 these 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; Liu and Tsang, 2016) in which the test at each decision node uses the linear combinations of features. Clearly, PDT is the general form of axis-parallel DT. This paper first studies two emerging challenges for decision trees – ultrahigh feature dimensionality and ultrahigh label dimensionality. Then, we develop budget-aware classifier (BAC) and supervised budgeted tree to address ultrahigh feature dimensionality. Under the BAC framework, we further propose a sparse coding tree to address the second challenge.

1.1 Challenges of Ultrahigh Feature Dimensionality

Recently, the emerging trends of ultrahigh feature dimensionality have been analyzed in Mitchell et al. (2004), Fan et al. (2009) and Zhai et al. (2014). For example, the Web continues to generate quintillion bytes of data daily (Zikopoulos et al., 2012; Zhai et al., 2014), leading to three key challenges for PDT classification on the webspam data set with 16,609,143 features (which is collected from Wang et al. (2012)) :

1. It is difficult to understand and interpret PDT on this data set. How to make PDT interpretable in ultrahigh dimensions poses a critical challenge;

2. Many features in high dimensional space are usually non-informative or noisy. Those noisy features will decrease the generalization performance of PDT and derogate from their promising results for dealing with non-linear separable data, especially when there are many noisy features;

3. It is desired to identify a small set of features that is able to represent the original feature space in this ultrahigh dimensional data set.

To overcome these challenges, it is imperative to develop a sparse PDT with respect to input features and its corresponding theory.

1.2 Challenges of Ultrahigh Label Dimensionality

Annotation, which aims to tag objects with some labels to enhance the semantic understanding of the objects, has arisen in various applications. For example, in image annotation (Boutell et al., 2004; Guillaumin et al., 2009), one needs to predict some relevant keywords, such as *beach*, *sky* and *tree*, to describe a natural scene image. When annotating documents

(Schapire and Singer, 2000; Paralic and Bednar, 2003), one may need to classify them into different groups according to their annotations, such as *News*, *Finance* and *Education*. In video annotation (Song et al., 2005; Tang et al., 2008), some labels, such as *Government*, *Policy* and *Election* are needed to describe the subject of the video. In these real-world applications, the object needs to be annotated with multiple labels. Thus, it can be formulated as a multi-label annotation problem (Tsoumakas et al., 2010; Chen and Lin, 2012; Deng et al., 2014; Zhang and Zhou, 2014; Liu and Tsang, 2015b; Koyejo et al., 2015; Yen et al., 2016), which aims to annotate multiple labels for each input instance.

One central challenging issue for practical multi-label annotation is scalability in ultrahigh label and feature dimensions. Suppose one needs to annotate the presence or absence of q labels for each testing instance with m features, early approaches, such as Tsoumakas et al. (2010) and Read et al. (2011), scaling the number of annotations to be linear with the number of labels, take $\mathcal{O}(q \times m)$ time for annotation. We cannot get the results of these exhaustive approaches on the large-scale data sets within one week, such as amazon data set with 2,812,281 labels and 337,067 features, which is collected from McAuley et al. (2015a,b).

Recently, many works (Tsoumakas et al., 2008; Agrawal et al., 2013; Prabhu and Varma, 2014) have put more effort in exploring tree-based algorithms to minimize the number of annotations for multi-label annotations with ultrahigh label dimensions. Among them, FastXML (Prabhu and Varma, 2014) is the most recently advanced technique, which has shown state-of-the-art rapid annotations. However, there is no generalized performance guarantee for FastXML and our extensive empirical study verifies that FastXML generally underperforms in terms of annotation performance. Moreover, some literature (Tsoumakas et al., 2008; Madjarov et al., 2012) has shown Homer (Tsoumakas et al., 2008) achieves good annotation performance. Unfortunately, Homer requires at least $\mathcal{O}(q^2)$ time for training. Thus, we cannot get the results of Homer even on medium-sized data sets within one week. The question is: can we design some efficient, yet accurate multi-label annotation algorithms in ultrahigh label and feature dimensions?

1.3 Our Contributions and Organization

To address these challenges, we first revisit the PDT model. Specifically, we first 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, and 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 budget-aware classifier (BAC) to perform feature selection and optimize training error loss and margin simultaneously, and then develop a supervised budgeted tree (SBT) to achieve non-linear 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

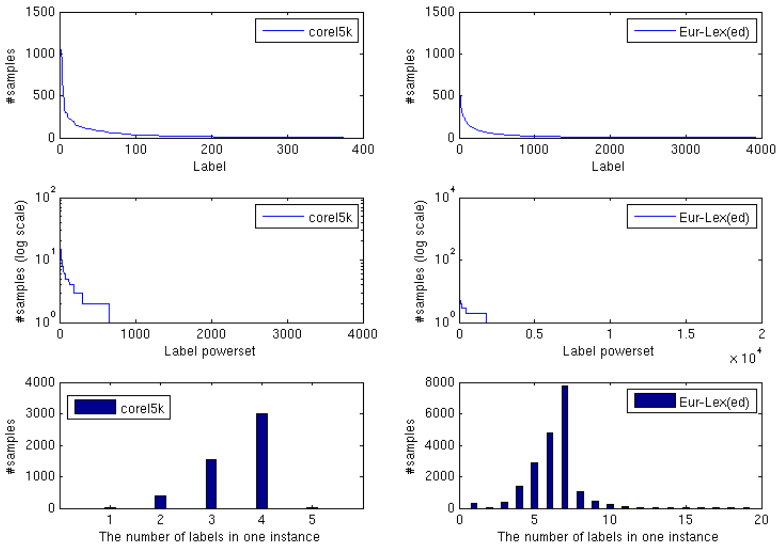

Figure 1: **Top**: Frequency of each label on the corel5k and Eur-Lex(ed) data sets. **middle**: Frequency of each label powerset on the corel5k and Eur-Lex(ed) data sets. **bottom**: Frequency of samples with the specific number of labels on the corel5k and Eur-Lex(ed) data sets.

minimizes the cross validation errors on each BAC. After that, we further diminish the generalization error by pruning the tree.

Based on BAC, we can design the tree-based algorithm to handle ultrahigh feature dimensions. How can one deal with ultrahigh label dimensions? To answer this question, we first use Figure 1 to demonstrate three important phenomena of real-world multi-label annotation data sets from various application domains[1]. We use corel5k and Eur-Lex(ed) data sets as examples. The details of these data sets are shown in Table 7.

Observations of Label Distribution: The labels of instances usually follow the distribution of Power Law (Barabási et al., 2000) which is shown in (Bhatia et al., 2015), as shown at the top of Figure 1, where many labels have very low label frequency. The bottom of Figure 1 shows the sparsity of labels in each instance, which means that the average number of labels in the data set is small. The middle of Figure 1 shows that very few label powersets have high frequency. Note that similar observations can also be found in image annotation tasks (Mao et al., 2013).

In real-world problems, although the number of labels would be very large, the first observation indicates that only a few labels appear very frequently; many others are rare. Based on this property, we design a tree-based algorithm to reduce the overall annotations as follows. Frequent labels should be predicted first in the decision tree, so there will be fewer annotations involved to predict the frequent labels, and more annotations for rare labels. To achieve this, we develop a novel annotation tree (AT) algorithm, which is analogous to the Shannon-Fano coding (Shannon, 1948) strategy, to deal with multi-label annotation problems. Moreover, the second observation shows that each instance typically

1. http://mulan.sourceforge.net

has few labels, which enables early stopping of our tree model. Thus, fewer annotations will be made along shorter paths in our tree model for most testing instances. Based on the property of the last observation, we also develop the powerset tree (PT) algorithm to deal with the transformed multi-class annotation task.

This paper is organized as follows. Section 2 reviews related work. Section 3 introduces a novel data-dependent generalization error bound for PDT. Based on our analysis, Section 4 develops the budget-aware classifier (BAC) to handle ultrahigh feature dimensions. Section 5 proposes the supervised budgeted tree (SBT) to construct a powerful non-linear classifier by arranging BAC in a tree-structure. Furthermore, based on BAC, Section 6 introduces the sparse coding tree framework for multi-label annotation with ultrahigh label and feature dimensions and provides generalization error bound for our proposed powerset tree (PT) and annotation tree (AT). Section 7 and Section 8 present the implementation issues and evaluate the performance of our proposed methods in ultrahigh dimensions, respectively. Lastly, Section 9 concludes the work.

2. Related Work

In this section, we introduce some backgrounds about the perceptron decision tree, multi-label annotation, multiple kernel learning, Huffman coding and Shannon-Fano coding.

2.1 Perceptron Decision Tree

OC1 (Murthy et al., 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 et al. (1994) show the competitive results of OC1 compared with C4.5 (Quinlan, 1993) (which is the famous axis-parallel DT) and other baselines. The authors in (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 from the margin only.

The combination of bagging and building ensembles of trees (Breiman, 2001) is a popular strategy to control the generalization error of decision trees. Moreover, due to PDT's performance is sensitive to the tree structure, which is determined by the distribution of features, ensembles offer an effective technique for obtaining increased accuracy by combining the predictions of many different trees, such as random forest feature selection (RFFS) (Hastie et al., 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 model (Zhu et al., 2003). Unfortunately, these l_1-regularized methods suffer from three major limitations:

1. Because the scale variation of weight parameters, it is non-trivial for these methods to control the number of features to be selected meanwhile to regulate the decision function;

2. Since l_1-norm regularization shrinks the regressors, the feature selection bias inevitably exists in the l_1-norm methods (Zhang and Huang, 2008; Zhang, 2010a,b);

3. l_1-norm regularized methods are inefficient or infeasible for handling the data sets with ultrahigh dimensions. Moreover, the theoretical and empirical analysis in Zhang and Huang (2008), Zhang (2010a), Zhang (2010b), and Tan et al. (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.

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. In this paper, we select GBFS and CSTC as the representatives of these kinds of methods for comparison.

2.2 Multi-Label Annotation

Much effort has been focused on annotation tasks, such as image annotation (Boutell et al., 2004) and video annotation (Song et al., 2005). Usually, annotation tasks can be formulated as a multi-label annotation problem (Tsoumakas et al., 2010; Chen and Lin, 2012; Deng et al., 2014; Zhang and Zhou, 2014; Koyejo et al., 2015; Liu and Tsang, 2015a; Gong et al., 2017). According to the annotation complexity, we divide these methods into several categories:

The first category includes some popular methods, such as Binary Relevance (BR) (Tsoumakas et al., 2010) and Classifier Chain (CC) (Read et al., 2011), the number of annotations scale linearly with the length of label vector. All these methods cannot handle ultrahigh label dimensions.

The second category includes the encoding-decoding strategy. For example, (Zhang and Schneider, 2011, 2012) first use different projection methods to transform the original label space into another space, and recover the original multiple labels using an expensive decoding process, which involves solving a quadratic programming (QP) problem on a space with a combinatorial nature.

The third category includes Label Powerset (LP) (Tsoumakas et al., 2010) and its variant (Tsoumakas and Vlahavas, 2007). LP reduces a multi-label annotation problem into a multi-class annotation problem by treating each distinct label set as one of the classes for a transformed multi-class learning task. One can then train a single multi-class classifier (Tsoumakas et al., 2010) or many binary classifiers (for example one-vs-all or one-vs-one) for the transformed multi-class annotation problem. The number of transformed classes

is upper bounded by $\min(n, 2^q)$ where n means the size of training data set and q means the number of labels. For large values of n and q, it imposes an extremely high training complexity.

The last category is tree-based algorithms, which have annotation costs that are logarithmic in the number of labels. FastXML (Prabhu and Varma, 2014) is one of the most advanced tree-based algorithms. However, there is no generalized performance guarantee for FastXML and our extensive empirical study verifies that FastXML generally underperforms in terms of annotation performance. Homer (Tsoumakas et al., 2008) is developed to use a divide-and-conquer-strategy to divide original problem into k sub-problems. Some literature (Tsoumakas et al., 2008; Madjarov et al., 2012) has shown Homer achieves good annotation performance. However, Homer requires at least $\mathcal{O}(q^2)$ time for training. Thus, we cannot get the results of Homer even on medium-sized data sets within one week.

2.3 Multiple Kernel Learning

By combining multiple sets of features, multiple kernel learning (MKL) (Lanckriet et al., 2004b; Rakotomamonjy et al., 2008) has become a powerful learning paradigm to enhance the interpretability of the decision function and improve performances of single kernel methods. MKL has been successfully applied in many tasks, such as object recognition (Bucak et al., 2014), signal processing (Subrahmanya and Shin, 2010), and genomic data fusion (Lanckriet et al., 2004a).

Many efficient MKL algorithms (Lanckriet et al., 2004b; Bach et al., 2004; Sonnenburg et al., 2006; Rakotomamonjy et al., 2008; Liu et al., 2015) are developed. SimpleMKL (Rakotomamonjy et al., 2008) is one of the most popular MKL solvers. SimpleMKL uses a sub-gradient method to solve the non-smooth optimization problem. However, it is expensive to calculate the sub-gradient for large-scale problems. Tan et al. (2014) develop an up-to-date MKL solver, which modifies accelerated proximal gradient method to solve the primal form of MKL and shows promising results. This paper will adapt this method to solve our subproblems.

2.4 Huffman Coding and Shannon-Fano Coding

Huffman coding (Huffman, 1952) is one of the most widespread bottom-up encoding algorithm for data compression. Huffman coding uses a specific method for choosing the representation for each symbol, resulting in a prefix code, that is, the bit string representing some particular symbol is never a prefix of the bit string representing any other symbol. Given a set of symbols and their weights (usually proportional to probabilities), Huffman coding aims to find a set of prefix codewords with minimum expected codeword length. Based on the frequency of occurrence of each symbol, the principle of Huffman coding is to use a lower number of bits to encode the symbol that occurs more frequently. The detailed procedure of Huffman coding can be referred to Huffman (1952). This work first uses the idea of Huffman coding to deal with the label powerset annotation problems.

Shannon-Fano coding (Shannon, 1948) is a top-down encoding technique for constructing a prefix code based on a set of symbols and their weights. Shannon-Fano coding arranges the symbols in order from biggest weight to smallest weight, and then divides them into two sets whose total weights are roughly comparable. Then, we encode symbols in the first

Table 1: Important notations used in this paper.

Notation	description
n	number of instances
m	dimensionality of the input space
q	dimensionality of the output space
$\mathcal{X}$	the input space over $\mathbb{R}^m$
$\mathcal{Y}$	the output space ($\mathcal{Y} = \{-1, 1\}$ for binary classification, while $\mathcal{Y} \subseteq \{1, 2, \cdots, q\}$ for multi-label classification)
x_i	$x_i \in \mathbb{R}^m$ is a real vector representing the i-th input or instance
y_i	$y_i \in \{-1, 1\}$ is used to represent the i-th label for binary classification
$\mathbf{y}_i$	$\mathbf{y}_i \in \{0, 1\}^q$ is used to represent the i-th label set for multi-label classification, where $\mathbf{y}_i(j) = 1$ if and only if $j \in \mathcal{Y}_i$
X	instance matrix, where $X = [x_1, \cdots, x_n]'$
Y	binary label matrix, where $Y = [y_1, \cdots, y_n]'$
ϱ	$\varrho = [\varrho_1, \cdots, \varrho_m]' \in [0, 1]^m$ represents a feature selection vector, and $\sqrt{\varrho}$ denotes the elementwise square root operator
B	budget constraint on the weight coefficients
$\mathcal{A}$	domain of dual variable α, where $\mathcal{A} = \{\alpha \mid \sum_{i=1}^n \alpha_i \geq 1, 0 \leq \alpha_i \leq C, i = 1, \cdots, n\}$
$\mathcal{E}$	domain of ϱ, where $\mathcal{E} = \{\varrho \in [0, 1]^m \mid \sum_{j=1}^m \varrho_j \leq B\}$

set as zero and symbols in the second set as one. As long as any sets with more than one member remain, the same process is repeated on these sets. The detailed procedure of Shannon-Fano coding can be referred to Shannon (1948). Analogous to the Shannon-Fano coding strategy, this work first proposes a novel annotation tree algorithm to deal with multi-label annotation problems.

3. Data-Dependent Generalization Error Bound for PDT

Inspired by Bartlett and Mendelson (2002) and 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. $||\cdot||$ denotes the l_2 norm. If $\mathcal{S}$ is a set, $|\mathcal{S}|$ denotes its cardinality. Let $\mathbb{R}$ represent real number, $\mathcal{X}$ be the input space over $\mathbb{R}^m$ 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 over n training samples $(x_1, y_1), \cdots, (x_n, y_n)$. Let F be a class of functions mapping from $\mathcal{X}$ to $\mathbb{R}$. We follow the definitions in Koltchinskii (2001), Bartlett and Mendelson (2002) and Mendelson (2002), and use $R_n(F)$ and $G_n(F)$ to denote the Rademacher and Gaussian complexity of F, respectively. The important notations in this paper are summarized in Table 1.

Definition 1 (Generalized Decision Trees (GDT), Bennett et al. (2000)) *Given $\mathcal{X} = \mathbb{R}^m$, 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 3 (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 $\phi(\alpha) \geq \Theta(-\alpha)$ and is Lipschitz with constant L, then with probability at least $1-\delta$ with respect to n training samples $(x_1, y_1), \cdots, (x_n, y_n)$ drawn independently according to D, every function in F satisfies*

$$P_D(y \neq sign(\mathfrak{F}(x))) \leq \hat{E}_n(\phi(y\mathfrak{F}(x))) + 2LR_n(F) + \sqrt{\frac{ln(2/\delta)}{2n}}$$

Theorem 4 (Shawe-Taylor and Cristianini (2004)) *Fix margin 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. Then, with probability at least $1-\delta$ with respect to n training samples $(x_1, y_1), \cdots, (x_n, y_n)$ drawn independently according to D, we have*

$$P_D(y \neq sign(g(x))) \leq \frac{1}{n\gamma}\sum_{i=1}^{n}\xi_i + \frac{4}{n\gamma}\sqrt{tr(K)} + 3\sqrt{\frac{ln(2/\delta)}{2n}}$$

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

We provide the data-dependent generalization error bound for PDT as follows:

Theorem 5 *Let T be a PDT, where the Euclidean norm of the linear functions in T is at most 1. Suppose that the depth of T is no more than d, and it contains no more than M leaves. Let $(x_1, y_1), \cdots, (x_n, y_n)$ be n training samples, 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 universal small 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 $(x_1^l, y_1^l), \cdots, (x_{n_l}^l, y_{n_l}^l)$ 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 3 implies that with probability at least $1 - \delta_1$:

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

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 $(x_1^l, y_1^l), \cdots, (x_{n_l}^l, y_{n_l}^l)$ 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 Theorem 5 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 4, 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 $1-\delta$:

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

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. ■

4. Budget-Aware Classifier

Theorem 5 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, that is $tr(K_l) \leq \mathfrak{B}$. Suppose there are n samples $(x_1, y_1), \cdots, (x_n, y_n)$, $x_i \in \mathbb{R}^m$ ($x_i = [x_{i,1}, \cdots, x_{i,m}]'$). Let $X = [x_1, \cdots, x_n]'$ and $Y = [y_1, \cdots, y_n]'$. 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: $\hat{x}_i = x_i \odot \sqrt{\varrho}$, where $\varrho = [\varrho_1, \cdots, \varrho_m]' \in [0,1]^m$ represents a feature selection vector and $\sqrt{\varrho}$ denotes the elementwise square root operator. 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 5, we hereinafter introduce the notion of budget-aware classifier (BAC).

Definition 6 (Budget-Aware Classifier (BAC)) *Assume there are n samples* $(x_1, y_1), \cdots, (x_n, y_n)$ *drawn independently according to a probability distribution D, where* $x_i \in \mathbb{R}^m$ *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:* $\hat{x}_i = x_i \odot \sqrt{\varrho}$ *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 budget-aware classifier contains a linear decision function* $g_w(x) = w'(x_i \odot \sqrt{\varrho} + b), ||w||^2 = 1$, *where the parameters are obtained by solving the following problem :*

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

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.*

4.1 Learning the BAC

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 problem (4) with respect to w, ξ, γ, 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 $\alpha_i, \psi_i, \nu \geq 0$. Differentiating with respect to the primal variables and substituting the relations obtained into the primal, we obtain the following dual objective function: $\mathcal{L}(\alpha, \lambda, \varrho) = -\frac{1}{4\lambda} \left\| \sum_{i=1}^n \alpha_i y_i (x_i \odot \sqrt{\varrho}) \right\|^2 - \lambda$. Optimizing λ gives $\lambda^* = \frac{1}{2} \| \sum_{i=1}^n \alpha_i y_i (x_i \odot \sqrt{\varrho}) \|$, resulting in $\mathcal{L}(\alpha, \varrho) = -\left\| \sum_{i=1}^n \alpha_i y_i (x_i \odot \sqrt{\varrho}) \right\|$. Optimizing the squared l_2-norm objective and l_2-norm objective yields the same solution. To simplify the optimization, we use the squared l_2-norm as the objective. Thus, problem (4) can be equivalently reformulated as the following problem:

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

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

Let $\mathbb{G}(\alpha, \varrho) = \frac{1}{2} \left\| \sum_{i=1}^n \alpha_i y_i (x_i \odot \sqrt{\varrho}) \right\|^2$. According to the minimax saddle-point theorem (Sion, 1958), we have the following theorem:

Theorem 7 *According to the minimax saddle-point theorem in (Sion, 1958), the following equality holds:*

$$\min_{\varrho \in \mathcal{E}} \max_{\alpha \in \mathcal{A}} -\mathbb{G}(\alpha, \varrho) = \max_{\alpha \in \mathcal{A}} \min_{\varrho \in \mathcal{E}} -\mathbb{G}(\alpha, \varrho)$$

Proof Recall that both $\mathcal{A}$ and $\mathcal{E}$ are convex compact set. It can be easily verified that the objective function is concave with respect to α. As shown in Section 4.1.1, the objective function is linear with respect to ϱ, and also is convex in ϱ. Therefore, the conditions in minimax saddle-point theorem in (Sion, 1958) are satisfied here. ■

Algorithm 1 Cutting Plane Algorithm for Solving Problem (7)

Input: Data set $\{x_i, y_i\}_{i=1}^n$. Initialize α_0.

1: $t = 0$, $\mathcal{C}_t = \emptyset$.
2: **repeat**
3: $t = t + 1$.
4: Worst-case analysis: Finding the most violated ϱ_t based on α_{t-1} and set $\mathcal{C}_t = \mathcal{C}_{t-1} \cup \{\varrho_t\}$. The details can be referred to Section 4.1.1.
5: Master-problem optimization: Solving problem (8) over $\mathcal{C}_t$ and obtaining the optimal solution α_t. The details can be referred to Section 4.1.2.
6: **until** ϵ-optimal

Based on Theorem 7, instead of directly solving problem (5), we address the following problem instead:

$$\min_{\alpha \in \mathcal{A}} \max_{\varrho \in \mathcal{E}} \mathbb{G}(\alpha, \varrho) \tag{6}$$

where we drop the negative sign of problem (6) for the clarity of our optimization procedure. By introducing variable $\theta \in \mathbb{R}$, problem (6) can be further formulated as a semi-infinite programming (SIP) (Pee and Royset, 2011) problem:

$$\min_{\alpha \in \mathcal{A}, \theta \in \mathbb{R}} \theta \;\; s.t. \;\; \theta \geq \mathbb{G}(\alpha, \varrho), \forall \varrho \in \mathcal{E} \tag{7}$$

Problem (7) involves infinite number of constraints. Inspired by the cutting plane algorithm (Kelley, 1960), which iteratively refines a feasible constraint set and then solves a sequence of approximating linear programs, we propose Algorithm 1 to solve problem (7). Specifically, given α_{t-1}, the worst-case analysis is to infer the most-violated ϱ_t, and add it into the active constraint set $\mathcal{C}_t$. Then, we update α_t by solving the following problem:

$$\min_{\alpha \in \mathcal{A}, \theta \in \mathbb{R}} \theta \;\; s.t. \;\; \theta \geq \mathbb{G}(\alpha, \varrho_h), \forall \varrho_h \in \mathcal{C}_t \tag{8}$$

The following convergence theorem indicates that Algorithm 1 gradually approaches to the optimal solution:

Theorem 8 *Algorithm 1 stops after a finite number of steps with the globally optimal solution of Problem (7).*

The proof can be adapted from Tan et al. (2014). Algorithm 1 involves two major steps: **worst-case analysis** and **master-problem optimization**, which will be discussed in the following subsections:

4.1.1 Worst-case analysis

The worst-case analysis is to solve the following maximization problem:

$$\max_{\varrho} \frac{1}{2} \Big\| \sum_{i=1}^{n} \alpha_i y_i (x_i \odot \sqrt{\varrho}) \Big\|^2 s.t. \sum_{j=1}^{m} \varrho_j \leq B, \varrho \in [0,1]^m \tag{9}$$

Note that $\left\| \sum_{i=1}^{n} \alpha_i y_i (x_i \odot \sqrt{\varrho}) \right\|^2 = \sum_{j=1}^{m} (\sum_{i=1}^{n} \alpha_i y_i x_{i,j})^2 \varrho_j$. Let $\Xi = (\sum_{i=1}^{n} \alpha_i y_i x_i)^2 \in \mathbb{R}^m$. Then, problem (9) can be transformed to $\max_{\varrho} \frac{1}{2} \sum_{j=1}^{m} \Xi_j \varrho_j$ subjected to $\sum_{j=1}^{m} \varrho_j \leq B, \varrho \in [0,1]^m$, which is a simple linear programming (LP) problem. We obtain the optimal solution for this LP problem by first sorting Ξ_j's and then setting the first B numbers corresponding to ϱ_j to one and the rests to zero. This procedure takes $\mathcal{O}(mlogB)$ time complexity, which is very efficient.

4.1.2 Master-problem optimization

Note that any $\varrho_h \in \mathcal{C}_t$ indexes a set of features. For simplicity, we define $X_h = [x_{1,h}, \cdots, x_{n,h}]' \in \mathbb{R}^{n \times B}$, where $x_{i,h}$ denotes the i-th instance with the features indexed by ϱ_h. By introducing dual variable $\mu_h \geq 0$ for each constraint defined by ϱ_h, the Lagrangian function of problem (8) can be written as: $\mathcal{L}(\theta, \alpha, \mu) = \theta + \sum_{\varrho_h \in \mathcal{C}_t} \mu_h (\mathbb{G}(\alpha, \varrho_h) - \theta)$. By setting its derivative with respect to θ to zero, we have $\sum \mu_h = 1$. By applying the minimax saddle-point theorem, problem (8) with the negative sign can be transformed to the following problem:

$$\max_{\alpha \in \mathcal{A}} \min_{\mu \in \mathcal{M}} - \sum_{\varrho_h \in \mathcal{C}_t} \mu_h \mathbb{G}(\alpha, \varrho_h) = \min_{\mu \in \mathcal{M}} \max_{\alpha \in \mathcal{A}} - \frac{1}{2} (\alpha \odot Y)' \Big(\sum_{\varrho_h \in \mathcal{C}_t} \mu_h X_h X_h^{'} \Big) (\alpha \odot Y) \tag{10}$$

where $\mathcal{M} = \{\mu | \sum \mu_h = 1, \mu_h \geq 0\}$.

Problem (10) is a multiple kernel learning (MKL) problem (Lanckriet et al., 2004b; Rakotomamonjy et al., 2008; Li et al., 2013), where the kernel matrix $\sum_{\varrho_h \in \mathcal{C}_t} \mu_h X_h X_h^{'}$ is a convex combination of $|\mathcal{C}_t|$ base kernel matrices $X_h X_h^{'}$. The resultant MKL problem can be solved efficiently by the modified accelerating proximal gradient (APG) method developed in (Tan et al., 2014), which takes $\mathcal{O}(n\varsigma B)$ time complexity, where ς is the number of the worst-case analyses. Thus, the time complexity of Algorithm 1 is $\mathcal{O}(n\varsigma B + mlogB)$, which is computationally efficient for large-scale and very high dimensional problems. Next sections will present what can we do with BAC.

5. Supervised Budgeted Tree

Due to the non-linear but highly interpretable representations, PDT has played a vital role in various data mining and machine learning applications. Unfortunately, PDT suffers from three limitations on ultrahigh dimensions which are described in introduction. To break the bottlenecks of PDT on ultrahigh dimensions, we first perform supervised learning based on our theoretical results and BAC, and construct a powerful non-linear classifier by arranging BAC in a tree-structure. In this section, we first present an algorithm for supervised budgeted tree (SBT), and propose pruning strategy to improve the generalization of SBT.

5.1 Algorithm for SBT

We build a SBT in a top-down approach. Starting from the root, we train the BAC on each node 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 2.

Algorithm 2 Supervised Budgeted Tree (SBT)

Input: Training data $(x_1, y_1), \cdots, (x_n, y_n)$, where $x_i \in \mathbb{R}^m$ and $y_i \in \{\pm 1\}$.
Output: A supervised budgeted tree

1: **while** BAC contains more than one class **do**
2: train the BAC on this node.
3: split the training data of this node into left child if these instances are predicted to be positive by the BAC and call Algorithm 2 with split data as the input.
4: split the rest of training data of this node into right child and call Algorithm 2 with the remaining data as the input.
5: **end while**
6: Return the complete supervised budgeted tree.

5.2 SBT Pruning

Theorem 5 reveals that the generalized bound also depends on the depth and leaves of the tree, which appear in the second 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 SBT. 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 these child nodes.

It is obvious that any slight changes in the data distribution result in generating different feature subsets for many feature selection methods, such as (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:

$$\min_{w^{\upsilon}, \varrho^{uni}} \sum_{\upsilon} BAC(w^{\upsilon}, \varrho^{uni}, S^{\upsilon}) \tag{11}$$

where S^{υ} represents the $\upsilon - th$ fold of training data, w^{υ} is the corresponding weight vector, and $BAC(w^{\upsilon}, \varrho^{uni}, S^{\upsilon})$ denotes problem (4). To solve problem (11), 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 et al., 2014) to get the w^{υ} for each data fold separately. We repeat these 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.

6. Sparse Coding Tree

In this section, based on BAC, we present sparse coding tree framework for multi-label annotation problem and the transformed multi-class (label powerset) annotation tasks.

Algorithm 3 Powerset Tree (PT) for Label Powerset Annotation

Input: Given ϖ transformed classes S, corresponding frequencies P, and $T_{s_i}(i = 1, \cdots, \varpi)$.
Output: Powerset Tree.

1: Initialize a priority queue $Q = \emptyset$ with the ascending order of the frequency.
2: Create a leaf node for each transformed class s_i and insert it to Q.
3: **while** $|Q| > 1$ **do**
4: remove the two nodes s_i and s_j of the lowest frequency from Q.
5: create a new internal node s_k with s_i and s_j as children and its frequency equals to the sum of the two nodes' frequency.
6: set the instance T_{s_i} as positive sample and T_{s_j} as negative sample, then construct a BAC for the node s_k.
7: insert the new node s_k to Q.
8: **end while**
9: The remaining node is the root node and the tree is complete.

6.1 Label Powerset Annotation

Our goal is to minimize the number of classifiers for binary decisions to enable rapid label powerset annotation for a number of testing instances. Recall that, to minimize the expected codeword length of information, Huffman coding uses shorter codewords to encode more frequent symbols and longer codewords to encode less frequent symbols. The idea of Huffman coding can be easily adapted to deal with the label powerset annotation problems.

Given a random n multi-labeled training sample, we first transform multi-label annotation to multi-class annotation by treating each unique set of labels as one of the classes for transformed multi-class learning task and record the frequency of each transformed class. Let ϖ be the number of transformed classes. A powerset tree is built based on the ϖ classes and their corresponding frequencies. Leaf nodes of this powerset tree are the set of transformed classes, denoted by $S = \{s_1, \cdots, s_\varpi\}$, and the corresponding frequencies are represented by $P = \{p(s_1), \cdots, p(s_\varpi)\}$. $T_{s_i}(i = 1, \cdots, \varpi)$ denotes the training instances associated with each transformed class. In each decision node, we construct a BAC and then build a powerset tree dealing with the label powerset case in a bottom-up manner. The details of the algorithm are presented in Algorithm 3.

More frequent classes are close to the root and less frequent classes are close to the leaf node layer. Thus, the annotation for most testing instances only require a few binary decisions along some shorter paths in our algorithm.

6.2 Multi-label Annotation

For multi-label annotation problems, the hypothesis space is exponential and we cannot observe some combinations of labels in the training data. Thus Huffman coding cannot be used to tackle the multi-label annotation problem. To reduce the number of annotations, we develop a novel annotation tree algorithm, which is analogous to the Shannon-Fano coding strategy. We first construct a BAC to predict the frequent label that have more examples. Then, we can leverage the annotations of such classifier to divide the remaining labels into two sets whose cardinality are roughly comparable. As long as any sets with

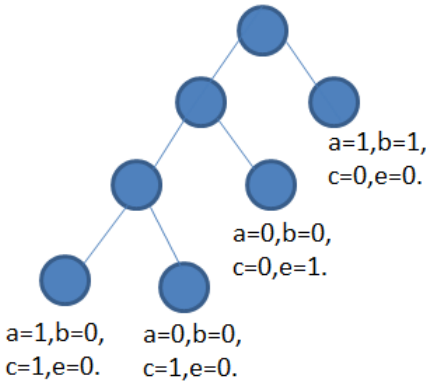

Figure 2: Schematic illustration of PT. We use a simple example to demonstrate our powerset tree. Assume one needs to annotate the presence or absence of four labels (a, b, c, e) for 8 input instances. Suppose $(1, 1, 0, 0)$, $(1, 0, 1, 0)$, $(0, 0, 0, 1)$ and $(0, 0, 1, 0)$ appear 3, 1, 3 and 1 times, respectively. Following Algorithm 3, we build the tree structure as shown in Figure 2.

more than one member remain, the same process is repeated on these sets. Our algorithm makes annotations for the ordered labels which accord to the arrangement with the label frequency ranging from high to low. The second observation shows that each instance has very few labels, based on our algorithm, so only a few annotations will be made for most testing instances. We start with some definitions associated with the specific nodes in a tree.

Let g represent the node in a tree, assume the label set associated with the node g is $G_g = \{1, 2, \cdots, q\}$, given a set of training instances $T_g = \{x_i\}_{i=1}^n$ and the corresponding labels $L_g = \{\mathbf{y}_i\}_{i=1}^n$.

Definition 9 *(Zero Label Set) refers to the situation where all the training instances at the node g do not belong to this label set. It defined as: $Zero_g = \{j|$ where all the training instances at the node g do not belong to label j $\}$.*

Definition 10 *(One Label Set) refers to the situation where all the training instances at the node g belong to this label set. It defined as: $One_g = \{j|$ where all the training instances at the node g belong to label j $\}$.*

To accelerate the learning process, these zero and one label sets will be thrown away. Thus we define the working label set for node g: $W_g = G_g - Zero_g - One_g$. Now, we introduce the learning process for the tree in three cases.

1. **Root node:** Assume g is the root node. We select label $l_g \in W_g$ with the highest frequency among W_g and build a BAC for l_g. g's left and right child are denoted as g_1 and g_0, respectively. The training instances of node g will be partitioned into left child g_1 if the values of label l_g of these instances is 1, otherwise the instances will be partitioned into right child g_0. The label set associated with node g_1 and g_0 will be

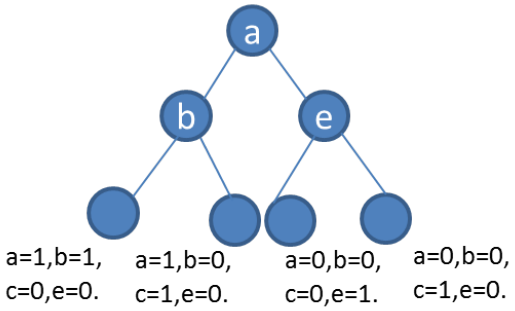

Figure 3: Schematic illustration of AT. We use the same example, which is aforementioned in Figure 2, to illustrate our annotation tree. Following Algorithm 4, we build a BAC for label a in the root node. For internal nodes, we build two BAC for label b and e in the left and right child nodes of the root, respectively. The working set in leaf nodes is empty, so we do not train any classifiers for the leaf nodes.

reduced as $W_g \backslash l_g$. The training instance sets for g_1 and g_0 are denoted by T_{g_1} and T_{g_0}.

2. **Internal node:** The training procedure for internal (child) nodes are the same with that for the root node but uses a subset of training instances. The training instances are split recursively into two smaller subsets until our algorithm moves down to the leaf nodes.

3. **Leaf node:** If the working set of the node is empty, then it is a leaf node and stop training for the leaf nodes.

We predict the multi-label output for a new testing instance from the root to leaves and assign the labels of the leaf node to this testing instance. The schematic illustration of our tree is shown in Figure 3 and the details of algorithm is presented as follows:

6.3 Testing Time Complexity Analysis and Generalization Error Bound

In this section, we study the testing time complexity and generalization error bound for both label powerset and multi-label annotations. We denote logarithms to base 2 by log. Let $\mathbf{T}$ and $\mathbb{T}$ represent PT and AT, respectively.

6.3.1 Testing Time Complexity Analysis

As the annotation efficiency critically depends on the number of annotation times, we first study the average number of annotations for the both cases.

We first introduce the noiseless coding theorem in (Roman, 1992). Let $C = \{c_1, \cdots, c_n\}$ be the symbol set of size n. $p(c_i)$ is the probability that symbol c_i occurs. After encoding of symbols in C, $l(c_i)$ denotes the length of code c_i and the average codeword length is defined as $AveLen(c_1, \cdots, c_n) = \sum_{i=1}^{n} p(c_i) l(c_i)$. The entropy of $\{p(c_1), \cdots, p(c_n)\}$ is de-

Algorithm 4 Annotation Tree (AT) for Multi-label Annotation

Input: Given node g and its corresponding G_g, T_g and L_g.
Output: Annotation Tree

Compute $Zero_g$, One_g and W_g
while $W_g \neq \emptyset$ **do**
  select label $l_g \in W_g$ with the highest frequency
  build a BAC for l_g
  set the label set of left and right child as $W_g \backslash l_g$
  split the training instances of node g into left child g_1 if the values of label l_g of these instances are 1 and call Algorithm 4 with input ($W_g \backslash l_g$, T_{g_1})
  split the rest of training instances of node g into right child g_0 and call Algorithm 4 with input ($W_g \backslash l_g$, T_{g_0})
end while
The tree is complete

fined as $H(p(c_1), \cdots, p(c_n)) = \sum_{i=1}^{n} p(c_i) \log(1/p(c_i))$. We get the following noiseless coding theorem:

Theorem 11 *(The Noiseless Coding Theorem) Given* $\{p(c_1),\cdots,p(c_n)\}$*, we have*

$$H(p(c_1), \cdots, p(c_n)) \leq AveLen(c_1, \cdots, c_n) < H(p(c_1), \cdots, p(c_n)) + 1$$

Label powerset annotation: We define $H(P_{lp}) = \sum_{i=1}^{\varpi} p(s_i) \log(1/p(s_i))$. $l(s_i)$ denotes the number of annotations for class s_i and the average number of annotations is defined as $mean(S) = \sum_{i=1}^{\varpi} p(s_i) l(s_i)$. Following Theorem 11, we obtain the bound of the average number of annotations for the label powerset annotation problem using Algorithm 3.

Theorem 12 *Assume we transform a random n multi-labeled sample to a multi-class sample which has* ϖ *classes. We obtain the bound of the average number of annotations for the transformed multi-class annotation problem based on* $\mathbf{T}$*:*

$$H(P_{lp}) \leq mean(S) < H(P_{lp}) + 1$$

Multi-label annotation: Assume there are Φ leaf nodes in a AT. These leaf nodes denoted by $S' = \{s'_1, \cdots, s'_\Phi\}$. Assume the frequency p'_i of each unique label set can be associated with leaf node s'_i. We define $H(P_l) = \sum_{i=1}^{\Phi} p(s'_i) \log(1/p(s'_i))$. Similar to label powerset annotation problem, we use $mean(S')$ to represent the average number of annotations of Algorithm 4. We obtain the bound of $mean(S')$ for multi-label annotation.

Theorem 13 *Given a random n multi-labeled sample which has* Φ *unique label set. We obtain the bound of the average number of annotations for multi-label annotation problem based on* $\mathbb{T}$*:*

$$H(P_l) \leq mean(S') < H(P_l) + 1$$

Table 2: Testing time complexity comparison between the main methods used in this paper. $\Upsilon, \Psi, \iota, \mathfrak{D}$: # clusters, the average # instances in each cluster, # learners and the dimension of the embedding space used in SLEEC. κ_w denotes the average number of non-zero weights of PD-Sparse.

Method	**Worst Testing Time**	**Average Testing Time**
BR	$\mathcal{O}(mq)$	$\mathcal{O}(mq)$
LP	$\mathcal{O}(m\vartheta)$	-
Homer	-	$\mathcal{O}(m\log_k(q))$
FastXML	-	$\mathcal{O}(\mathcal{T}m\log(q))$
SLEEC	-	$\mathcal{O}(\mathfrak{D}\Upsilon\iota + \mathfrak{D}\Psi\iota + m\mathfrak{D})$
PD-Sparse	-	$\mathcal{O}(\kappa_w q)$
PT	$\mathcal{O}(\varsigma B\log(\vartheta))$	$\mathcal{O}(\varsigma BH(P_{lp}))$
AT	$\mathcal{O}(\varsigma Bq)$	$\mathcal{O}(\varsigma BH(P_l))$

Proof Assume there are n multi-labeled samples which have Φ unique label set. We build a hierarchical tree based on n sample according to Algorithm 4. Each node is associated with one label, so the labels associated with nodes in each path of the tree are one of the unique label set, so there are Φ number of paths and leaf nodes. The frequency of each unique label set can be associated with each leaf node. After this transformation, we apply Theorem 12 to obtain the above bounds. ■

Therefore, in the label powerset and multi-label annotation algorithms, we can expect that the number of annotations for each instance are around $H(P_{lp})$ and $H(P_l)$, which are much fewer than $\min(n, 2^q)$ and q, respectively.

Algorithm 4 and Algorithm 3 are denoted as AT and PT, respectively. We provide the testing time complexity analysis for each testing instance. The number of annotations for BR are equal to the number of labels. Each time Homer divides the problem into k sub-problems. Following (Tsoumakas et al., 2008), it takes $\mathcal{O}(m\log_k(q))$ on testing time. According to (Prabhu and Varma, 2014), the average cost of prediction for FastXML is $\mathcal{O}(\mathcal{T}m\log(q))$, where $\mathcal{T}$ is the number of trees in the FastXML ensemble. In the worst case, LP requires $\min(n, 2^q)$ times of annotations respectively. The number of annotations for PT and AT are $\log(\min(n, 2^q))$ and q, respectively. As shown in our theoretical analysis, on average, the number of annotations can be further reduced to $H(P_{lp})$ and $H(P_l)$. The testing time complexity for the main methods used in this paper is presented in Table 2, where $\vartheta = \min(n, 2^q)$. From Table 2, we can see that the average testing time cost of our proposed PT and AT is lower than other methods.

6.3.2 Generalization Error Bound

Label powerset annotation: This subsection aims to bound the generalization error for a specific transformed class k in terms of the nodes in which instances in class k are

used to train the classifiers. We define the probability of error for transformed class k as $\epsilon_k(\mathbf{T}) = P\{x : x \in \mathbb{R}^m, x \text{ in class } k \text{ and } x \text{ is misclassified by } \mathbf{T} \text{ or } x \text{ is misclassified as class } k \text{ by } \mathbf{T}\}$. We first state some important definition, lemma and theorem.

We begin with the definition of the fat shattering dimension.

Definition 14 (Kearns and Schapire (1990)) *Let $\mathcal{H}$ be a set of real valued functions. We say that a set of points $\mathfrak{P}$ is γ-shattered by $\mathcal{H}$ relative to $r = (r_p)_{p\in\mathfrak{P}}$ if there are real numbers r_p indexed by $p \in \mathfrak{P}$ such that for all binary vectors b indexed by $\mathfrak{P}$, there is a function $f_b \in \mathcal{H}$ satisfying*

$$f_b(p) = \begin{cases} \geq r_p + \gamma & \text{if } b_p = 1 \\ \leq r_p - \gamma & \text{otherwise} \end{cases}$$

The fat shattering dimension $fat(\gamma)$ of the set $\mathcal{H}$ is a function from the positive real numbers to the integers which maps a value γ to the size of the largest γ-shattered set, if this is finite, or infinity otherwise.

Then, we provide the following lemma and theorem:

Lemma 15 *Given a specified $\mathbf{T}$ model with K decision nodes with margins $\gamma^1, \gamma^2, \cdots, \gamma^K$ at the decision nodes satisfying $k_i = fat(\gamma^i/8)$, where fat is continuous from the right. If $\mathbf{T}$ has correctly classified n multi-labeled examples $\boldsymbol{s}$ generated independently according to the unknown (but fixed) distribution D and $\bar{\boldsymbol{s}}$ is a set of another n multi-labeled examples, then we can bound the following probability to be less than δ: $P\{\boldsymbol{s}\bar{\boldsymbol{s}} : \exists\ \mathbf{T}$, it correctly classifies $\boldsymbol{s}$, fraction of $\bar{\boldsymbol{s}}$ misclassified $> \epsilon(n, K, \delta)\} < \delta$, where $\epsilon(n, K, \delta) = \frac{1}{n}(\mathfrak{Q}\log(8n) + \log\frac{2^K}{\delta})$ and $\mathfrak{Q} = \sum_{i=1}^{K} k_i \log(\frac{4en}{k_i})$.*

The proof can be adapted from Shawe-Taylor and Cristianini (1997).

Theorem 16 (Bartlett and Shawe-Taylor (1998)) *Let $\mathcal{H}$ be restricted to points in a ball of $\mathbf{M}$ dimensions of radius R about the origin, then*

$$fat_{\mathcal{H}}(\gamma) \leq \min\left\{\frac{R^2}{\gamma^2}, \mathbf{M} + 1\right\} \tag{12}$$

Based on Lemma 15 and Theorem 16, we derive the following theorem:

Theorem 17 *Assume we transform a random n multi-labeled sample to multi-class sample which has ϖ classes. Suppose transformed class k can be correctly classified by a $\mathbf{T}$, where the internal nodes are K decision nodes with margins γ^i at node i. In addition, suppose the number of nodes in which instances in class k are used to train the classifiers is $\mathbf{N}(k)$. Then with probability greater than $1 - \delta$,*

$$\epsilon_k(\mathbf{T}) \leq \frac{130R^2}{n}(D'_k \log(4en)\log(4n) + \mathbf{N}(k)\log(2n) + \log\frac{2}{\delta})$$

where $D'_k = \sum_{i\in k-nodes}\frac{1}{(\gamma^i)^2}$ (k-nodes are the set of nodes in which instances in class k are used to train the classifiers) and R is the radius of a ball containing the distributions support.

The proof can be adapted from Platt et al. (1999).

We discuss the implications of Theorem 17. Theorem 17 reveals one important factors to reduce generalization error bound for a specific transformed class k: it is to control the complexity of the model by reducing the average number of decision nodes. We build a **T** for the transformed multi-class classification problem. So, we obtain shorter path for higher frequency classes than that for less frequent classes. Based on Theorem 12, we can see that the average number of annotations of Algorithm 3 is around $H(P_{lp})$. Thus, PT is able to reduce the generalization error.

Multi-label annotation: This subsection analyzes the generalization error bound of multi-label annotation problem using Algorithm 4. Let $Path_{k-nodes}$ be the set of decision nodes along k-th path in $\mathbb{T}$. Assume $\epsilon_k(\mathbb{T})$ is the probability that any of node in $Path_{k-nodes}$ makes mistake. We provide the following generalization error bound for $\epsilon_k(\mathbb{T})$:

Theorem 18 *Given random n multi-labeled sample, suppose there is no mistake made by any node in $Path_{k-nodes}$ of $\mathbb{T}$, where the internal nodes are K decision nodes with margins γ^i at node i. Denote $\mathbb{N}(k) = |Path_{k-nodes}|$, that is the number of decision nodes along k-th path in $\mathbb{T}$. Then with probability greater than $1-\delta$,*

$$\epsilon_k(\mathbb{T}) \leq \frac{130R^2}{n}(D'_k \log(4en)\log(4n) + \mathbb{N}(k)\log(2n) + \log\frac{2}{\delta})$$

where $D'_k = \sum_{i \in Path_{k-nodes}} \frac{1}{(\gamma^i)^2}$ and R is the radius of a ball containing the distributions support.

The proof can be adapted from Bennett et al. (2000).

We discuss the implications of Theorem 18. Theorem 18 reveals one important factors to reduce generalization error bound for some decision nodes along a specific path in $\mathbb{T}$: it is to control the complexity of the model by reducing the average number of decision nodes. Based on Theorem 13, we can see that the average number of annotations of Algorithm 4 is around $H(P_l)$. Thus, our proposed AT is able to tighten the generalization error bound.

7. Practical Issues

Accessing features in sparse format is very expensive for high dimensional data sets. This paper proposes two efficient auxiliary data structures to reduce the computation cost of indexing and accessing features. We first propose a modified direct indexing method for small and medium-sized data sets.

Let ε be the selected feature set of the tree and $\mathcal{Q}_{|\varepsilon|}$ represent the natural number set which contains $|\varepsilon|$ numbers from 0 to $|\varepsilon|-1$. Then, we build an injection table: $\mathcal{I} : \varepsilon \mapsto \mathcal{Q}_{|\varepsilon|}$ to compress the whole selected feature set to a continuous index set. To support direct indexing for every instance, we maintain an array with size $|\varepsilon|$. For each feature that belongs to ε, we use $\mathcal{I}$ to find out the index number, and store the feature value at that index in the array.

To handle very high dimensional data sets, in which the data is usually stored in sparse format, we maintain a hash table to store the feature ID as the key and the feature value for each instance. The method takes $\mathcal{O}(1)$ time to access the features.

Table 3: Data sets used for SBT

Data Set	# 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	15,997	3,999
webspam	16,609,143	28,000	7,000

8. Experiment

In this section, we conduct some experiments to evaluate the performance of our proposed methods.

8.1 Evaluation on SBT

In this subsection, 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 LIBSVM website[2]. pcmac data set is from (Xu et al., 2014). The training/testing partition is either predefined or the data is randomly split into 80% training and 20% testing. The statistics of these data sets are described in Table 3.

We compare SBT with a number of baseline methods: standard SVM (l_2-SVM)[3], CART-LC (Breiman et al., 1984), OC1 (Murthy et al., 1994), LDKL (Jose et al., 2013), GBFS (Xu et al., 2014), CSTC (Xu et al., 2013) and FGM (Tan et al., 2014). We use the linear classification/regression package LIBLINEAR (Fan et al., 2008) with L2-regularized square hinge loss (primal) to implement standard SVM. LDKL uses composite non-linear kernels, and achieves significantly better classification accuracies as compared to localized multiple kernel learning (Gönen and Alpaydin, 2008) (which is competitive with RBF-SVM) and other state-of-the-art non-linear methods. Therefore, we do not compare with other non-linear SVMs. The codes of other baseline methods are provided by their authors.

We use 5-fold cross validation to prune SBT. Following the parameter settings in (Tan et al., 2014), B is chosen in a range of $\{2, 5, 10, 20, 50, 100, 150, 200, 250, 400\}$ for rcv, news20 and webspam data sets, 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 other data sets. Following the settings in (Oiwa and Fujimaki, 2014), the tree-depth is fixed to 3 in LDKL.

8.1.1 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 et al., 2014), we generate the synthetic data as follows: At first, we generate 10,000 training

2. https://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/
3. https://www.csie.ntu.edu.tw/∼cjlin/liblinear/

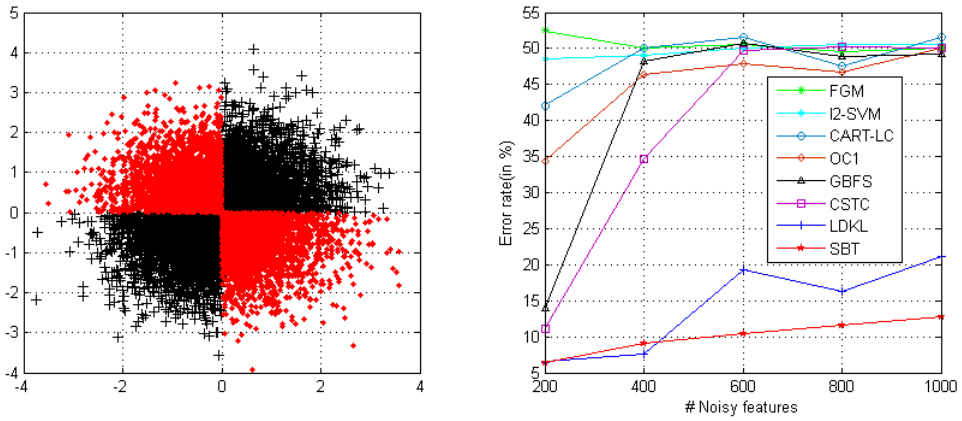

Figure 4: Testing error rates (in %) on the XOR synthetic data with different # noisy features.

instances, 2,000 testing instances and 200 features. The first two dimensions are informative features, which are independently sampled from Gaussian distribution $\mathcal{N}(0, 1)$. The remaining dimension are noisy features, which are independently sampled from Uniform distribution $\mathcal{U}(0, 1)$. The output y is decided by these 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 data set and test whether the considered methods can successfully identify the former two informative features and have stable and accurate classification performance. B is chosen in a range of $\{1, 3, 5, 10\}$ using cross validation for SBT. Figure 4 (left panel) shows two informative features.

Figure 4 (right panel) shows that: 1) l_2-SVM cannot work on the XOR data set. 2) FGM has shown impressive results in (Tan et al., 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 number of noisy features. 4) LDKL and SBT are most successful, while our model generates more stable and accurate results.

8.1.2 Pruning Performance

To evaluate the pruning performance, this subsection compares the classification performance of our method with pruning and without pruning on epsilon and rcv data sets. The results are shown in Table 4. From this table, we can see that our proposed pruning strategy significantly improves the generalization performance, which corroborates our theoretical findings.

Table 4: Testing error rates (in %) for our method with pruning and without pruning. The best results are in bold.

Data set	**SBT without pruning**	**SBT with pruning**
epsilon	17.23	**12.45**
rcv	6.19	**3.85**

Table 5: Testing error rates (in %) for different methods. Numbers in parentheses indicate the depth of tree. The best results are in bold.

Data set	**l_2-SVM**	**CART-LC**	**OC1**	**LDKL**	**SBT**
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)

8.1.3 Classification Performance

To verify the effectiveness of conducting feature selection in the decision tree, we evaluate the classification performance of SBT on real-world data sets compared with l_2-SVM, CART-LC, OC1 and LDKL. The results are shown in Table 5.

The results of Table 5 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) SBT 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) SBT has the best performance compared with baselines, while generating a small tree.

8.1.4 Feature Selection Performance

We evaluate the feature selection performance of SBT 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 5 show that: 1) FGM and SBT 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) SBT outperforms FGM. Thus, it is imperative to use the tree-structure to conduct feature selection.

Table 6: Training time (in second) for GBFS and our method on rcv data set. The faster results are in bold.

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

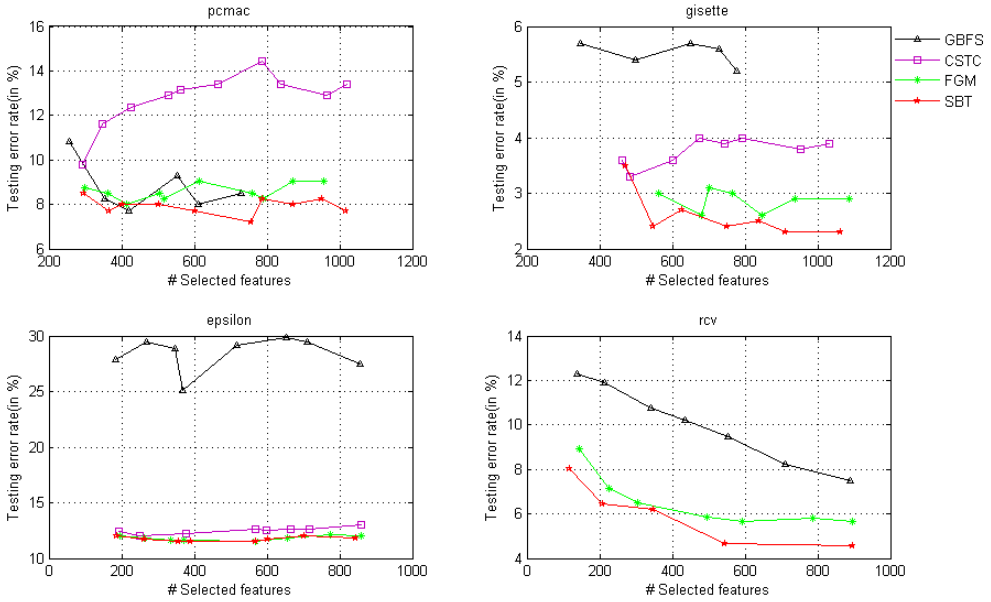

Figure 5: Testing error rates (in %) vs. # selected features for different methods on pcmac, gisette, epsilon and rcv data sets.

We next use rcv data set for training time comparison. The results are shown in Table 6. Since CART-LC, OC1 and CSTC are very slow on this data set, we do not report their training time here. From Table 6, 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 8019.3 seconds(s).

8.1.5 Results on Very High Dimensional Data Sets

We evaluate our method on news20 and webspam data sets with millions of dimensions. As CART-LC, OC1, LDKL, GBFS and CSTC are very slow on these data sets, we compare our method with l_2-SVM and FGM only on these data sets. The results are reported in

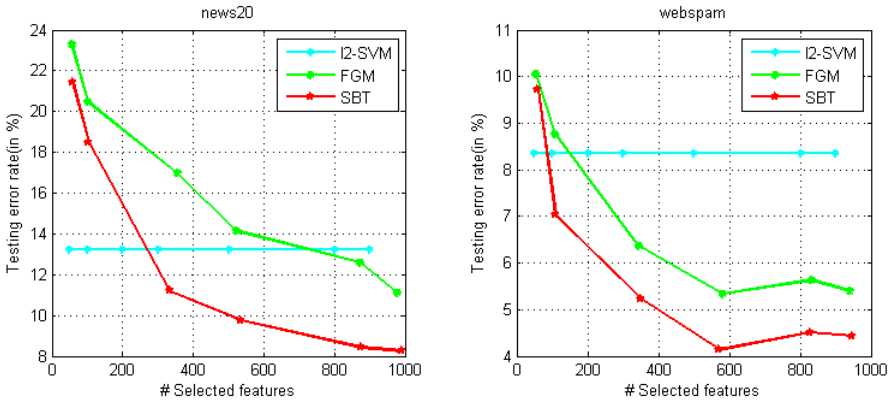

Figure 6: Testing error rates (in %) vs. # selected features for l_2-SVM, FGM and our method on news20 and webspam data sets. l_2-SVM uses all features.

Figure 6. From Figure 6, we can see that SBT consistently outperforms FGM and l_2-SVM on very high dimensional data sets.

The representation of SBT ($B = 2$) on webspam data set with 16,609,143 features is shown in Figure 7. In each decision node of Figure 7, we can see that our BAC can identify more informative and compact features. Thus, our SBT is easy to understand and interpret in ultrahigh dimensions.

8.2 Evaluation on PT and AT

In this subsection, we evaluate the performance of our proposed algorithms for multi-label annotation on a variety of real-world data sets from different application domains, which are collected from website[4][5]. The data sets fall into two categories. The first category contains five small and medium-sized data sets. The second category contains three large-scale data sets with very high label and feature dimensions. The details of data sets are shown in Table 7.

We compare our algorithms with several state-of-the-art multi-label annotation methods: 1) Flat methods: BR and LP; 2) Tree-based methods: Homer and FastXML; 3) Embedding-based method: Sparse local embedding for extreme classification (SLEEC) (Bhatia et al., 2015) is one of the most advanced embedding based method. 4) Sparse model: PD-Sparse (Yen et al., 2016) is a very recently proposed multiclass and multi-label approach which uses L1 regularization. The codes are provided by the respective authors. According to (Tsoumakas et al., 2008), k is chosen in a range of $\{2, 3, \cdots, 8\}$ using cross validation for Homer. Following (Prabhu and Varma, 2014), $\mathcal{T}$ is fixed to 50 for FastXML. We use the linear classification/regression package LIBLINEAR (Fan et al., 2008) with L2-regularized square hinge loss (primal) to train the classifier for both BR and LP. We use the default parameter in LIBLINEAR. According to the original setting in (Bhatia et al., 2015), we set

4. http://mulan.sourceforge.net
5. http://manikvarma.org/#Julian15a

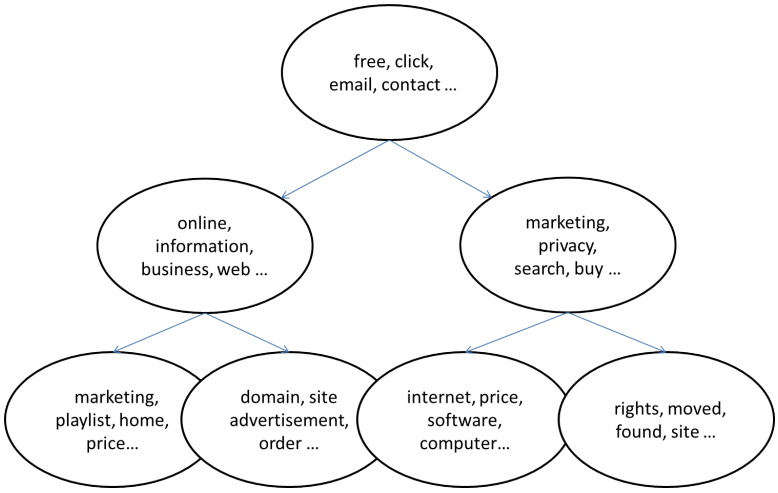

Figure 7: Representation of SBT on webspam data set.

Table 7: Data sets used for PT and AT

Data Set	# Labels	# Features	# Instances	Domain
mediamill	101	120	43,907	video
cal500	174	68	502	music
corel5k	374	499	5,000	image
Eur-Lex (dc)	412	5,000	19,348	text
Eur-Lex (ed)	3,993	5,000	19,348	text
wiki	30,938	101,938	20,762	web
delicious	205,443	782,585	26,701	web
amazon	2,812,281	337,067	10,482	advertisement

the number of the clusters as $\lfloor n/6000 \rfloor$, the number of learners as 15 and the dimension of the embedding space as 50 for small and medium-sized data sets, and 100 for very high dimensional data sets. We set B according to parameter settings in Section 8.1.

To measure the annotation performance of our methods and baseline methods fairly, this paper considers the standard F1 evaluation measurement (Du et al., 2017), which computes the F1 score for all the labels of each testing sample and takes the average over these samples. We use the publicly available split of the training and testing sets for very high dimensional data sets and perform 10-fold cross-validations on small and medium-sized data sets and report the mean and standard error of the F1 measure.

8.2.1 Results on Small and Medium-Sized Data Sets

This subsection studies the performance of various methods on mediamill, cal500, corel5k, Eur-Lex (dc) and Eur-Lex (ed) data sets. In our experiment, we cannot get the results of LP and Homer on Eur-Lex (ed) data set within one week. The F1 results of various methods on small and medium-sized data sets are listed in Table 8. From this table, we observe that: 1) Tree-based methods, such as Homer and FastXML, generally underperforms on all data sets.

Table 8: F1 Results of various methods on small and medium-sized data sets (mean ± standard deviation). The best results are in bold. Numbers in square brackets indicate the rank. "-" indicates that we cannot get the results of baselines within one week.

Data Set	BR	LP	Homer	FastXML	SLEEC	PD-Sparse	PT	AT
mediamill	0.414±0.00[3]	0.394±0.01[8]	0.411±0.00[4]	0.408±0.00[7]	0.411±0.00[4]	0.409±0.00[6]	0.422±0.01[2]	**0.462**±0.00[1]
cal500	0.317±0.02[6]	0.345±0.02[2]	0.321±0.01[5]	0.259±0.01[8]	0.310±0.03[7]	0.322±0.01[4]	0.324±0.02[3]	**0.362**±0.01[1]
corel5k	0.100±0.02[6]	0.138±0.01[2]	0.113±0.01[4]	0.056±0.01[8]	0.084±0.01[7]	0.110±0.02[5]	0.123±0.01[3]	**0.141**±0.01[1]
EUR-Lex(dc)	0.713±0.01[5]	0.659±0.01[6]	0.625±0.01[7]	0.516±0.01[8]	0.716±0.01[3]	0.718±0.01[2]	0.715±0.01[4]	**0.727**±0.01[1]
EUR-Lex(ed)	0.268±0.01[5]	-	-	0.254±0.01[6]	0.281±0.01[3]	0.302±0.01[2]	0.274±0.01[4]	**0.341**±0.01[1]
Ave. Rank	5.0	4.5	5.0	7.4	4.8	3.8	3.2	1.0

Table 9: F1 results of various methods on very high dimensional data sets. The best results are in bold.

Data Set	FastXML	SLEEC	PD-Sparse	PT	AT
wiki	0.1272	0.1364	0.1395	0.1422	**0.1657**
delicious	0.0077	0.0095	0.0086	0.0111	**0.0157**
amazon	0.0023	0.0148	0.0151	0.0166	**0.0348**

2) SLEEC, which is one of the most advanced embedding method, generally underperforms on small data sets, while it obtains competitive results on medium-sized data sets, such as Eur-Lex (dc) and Eur-Lex (ed). 3) PT and AT are two most successful methods, which significantly outperform tree-based algorithms, such as Homer and FastXML. The results verify our theoretical analysis.

Figure 8 shows the testing time of various methods spent on all testing instances per data set. From Figure 8.(a), we observe that: 1) SLEEC is slower than other baselines on all data sets. 2) Tree-based method, such as FastXML, is faster than BR, LP and SLEEC on medium-sized data sets. 3) PT and AT are faster than other baselines on all small and medium-sized data sets. For example, PT obtains around sixty times speedup over BR and LP, and about ten times speedup over FastXML on medium-sized data sets, such as Eur-Lex (dc). The results verify our testing time complexity analysis.

We next use Eur-Lex (ed) data set for training time comparison. BR, FastXML, SLEEC and PD-Sparse take 37652.1s, 269.9s, 5230.8s and 451.3s, respectively, while PT and AT take 839.8s and 1088.6s, respectively. Because BR and SLEEC require an expensive training process for each label and time-consuming embedding procedure, respectively, our proposed tree-based algorithms are able to achieve faster training than BR and SLEEC.

8.2.2 Results on Very High Dimensional Data Sets

In this subsection, we evaluate the performance of various methods on three very high dimensional data sets: wiki, delicious and amazon. In our experiment, we cannot get the

Table 10: F1@5 results of FastXML, SLEEC, PT and AT on wiki, delicious and amazon data sets. The best results are in bold.

Data Set	FastXML	SLEEC	PT	AT
wiki	0.5487	0.6067	0.6211	**0.6284**
delicious	0.4442	0.4631	0.4695	**0.4711**
amazon	0.4906	0.5077	0.5143	**0.5296**

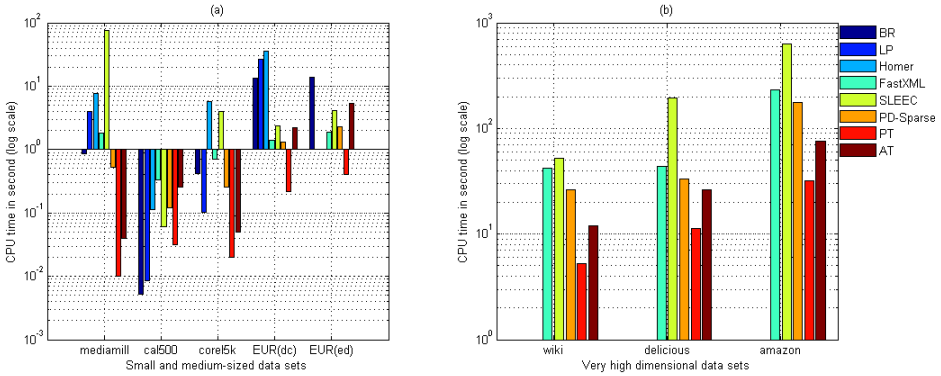

Figure 8: Testing time of various methods on all data sets (EUR-Lex is abbreviated to EUR).

results of BR, LP and Homer on these data sets within one week. The F1 results of various methods on three very high dimensional data sets are listed in Table 9. From Table 9 and Figure 8.(b), we can see that: 1) SLEEC outperforms FastXML, while FastXML is faster than SLEEC. 2) PD-Sparse is faster and more accurate than SLEEC and FastXML, which is consistent with the empirical studies in (Yen et al., 2016). 3) PT and AT are several times faster than FastXML, SLEEC and PD-Sparse. Moreover, PT and AT are much more accurate than FastXML, SLEEC and PD-Sparse on the data sets with millions of labels and features. Finally, we conclude that our proposed PT and AT are effective and efficient in ultrahigh label and feature dimensions.

Using Precision@5 and Recall@5, Table 10 shows the F1@5 results of FastXML, SLEEC, PT and AT on wiki, delicious and amazon data sets. From this table, we can see that our methods also outperform FastXML and SLEEC in terms of F1@5 measurement.

9. Conclusion

In this paper, we first develop a novel data-dependent generalization error bound for the PDT, where the data dependency of the bound comes from the training error loss, the

margin, the capacity of the kernel matrix defined on the training data, and the complexity of the tree. It provides the theoretical justification to learn a sparse linear hyperplane in each decision node and to prune the SBT. Our analysis reveals a new insight for the design of decision tree algorithms. Based on BAC and three important phenomena of real-world data sets from a variety of application domains, we develop the sparse coding tree framework for multi-label annotation with ultrahigh label and feature dimensions and provide generalized performance guarantee for our proposed PT and AT. Compared with state-of-the-art baselines, the results on the synthetic data set show that SBT is more resilient to noisy features, and empirical studies on real-world data sets demonstrate that SBT is easier to understand and interpret in ultrahigh dimensions and can identify more informative features than state-of-the-art feature selection methods. Moreover, our proposed PT and AT algorithms are able to annotate as well as possible, without performing unnecessary annotations in ultrahigh label and feature dimensions.

Acknowledgments

We would like to thank the action editor and reviewers for their valuable comments and constructive suggestions on our paper. This project is supported by the ARC Future Fellowship FT130100746, ARC grant LP150100671, DP170101628, DP150102728 and DP150103071.

References

Rahul Agrawal, Archit Gupta, Yashoteja Prabhu, and Manik Varma. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *WWW*, pages 13–24, May 2013.

Francis R. Bach, Gert R. G. Lanckriet, and Michael I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *ICML*, page 6, 2004.

A.-L. Barabási, R. Albert, H. Jeong, and G. Bianconi. Power-law distribution of the world wide web. *Science*, 287(54561):2115, 2000.

Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *JMLR*, 3:463–482, 2002.

Peter L. Bartlett and John Shawe-Taylor. Generalization performance of support vector machines and other pattern classifiers. In *Advances in Kernel Methods - Support Vector Learning*, pages 43–54. MIT Press, Cambridge, MA, USA, 1998.

Kristin P. Bennett, Nello Cristianini, John Shawe-Taylor, and Donghui Wu. Enlarging the margins in perceptron decision trees. *Machine Learning*, 41(3):295–313, 2000.

Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. Sparse local embeddings for extreme multi-label classification. In *NIPS*, pages 730–738, 2015.

Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771, 2004.

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

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

Serhat Selcuk Bucak, Rong Jin, and Anil K. Jain. Multiple kernel learning for visual object recognition: A review. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(7):1354–1369, 2014.

Yao-Nan Chen and Hsuan-Tien Lin. Feature-aware label space dimension reduction for multi-label classification. In *NIPS*, pages 1538–1546, 2012.

Marcin Czajkowski, Marek Grzes, and Marek Kretowski. Multi-test decision tree and its application to microarray data classification. *Artificial Intelligence in Medicine*, 61(1): 35–44, 2014.

Jia Deng, Olga Russakovsky, Jonathan Krause, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. Scalable multi-label annotation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3099–3102, 2014.

Bo Du, Mengfei Zhang, Lefei Zhang, Ruimin Hu, and Dacheng Tao. PLTD: patch-based low-rank tensor decomposition for hyperspectral images. *IEEE Trans. Multimedia*, 19(1): 67–79, 2017.

Jianqing Fan, Richard Samworth, and Yichao Wu. Ultrahigh dimensional feature selection: Beyond the linear model. *JMLR*, 10:2013–2038, 2009.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Lin Chih-Jen. Liblinear: A library for large linear classification. *JMLR*, 9:1871–1874, 2008.

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

Mehmet Gönen and Ethem Alpaydin. Localized multiple kernel learning. In *ICML*, pages 352–359, 2008.

Chen Gong, Dacheng Tao, Wei Liu, Liu Liu, and Jie Yang. Label propagation via teaching-to-learn and learning-to-teach. *IEEE Trans. Neural Netw. Learning Syst.*, 28(6):1452–1465, 2017.

Matthieu Guillaumin, Thomas Mensink, Jakob Verbeek, and Cordelia Schmid. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *ICCV*, pages 309–316, 2009.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. New York: Springer New York Inc, 2001.

David A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the Institute of Radio Engineers*, 40(9):1098–1101, 1952.

Cijo Jose, Prasoon Goyal, Parv Aggrwal, and Manik Varma. Local deep kernel learning for efficient non-linear svm prediction. In *ICML*, pages 486–494, 2013.

Michael J. Kearns and Robert E. Schapire. Efficient distribution-free learning of probabilistic concepts. In *Proceedings of the 31st Symposium on the Foundations of Computer Science*, pages 382–391, 1990.

J. E. Kelley. The cutting plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics*, 8(4):703–712, 1960.

Vladimir Koltchinskii. Rademacher penalties and structural risk minimization. *IEEE Trans. Information Theory*, 47(5):1902–1914, 2001.

Oluwasanmi Koyejo, Nagarajan Natarajan, Pradeep Ravikumar, and Inderjit S. Dhillon. Consistent multilabel classification. In *NIPS*, pages 3321–3329, 2015.

Gert R. G. Lanckriet, Tijl De Bie, Nello Cristianini, Michael I. Jordan, and William Stafford Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20(16):2626–2635, 2004a.

Gert R. G. Lanckriet, Nello Cristianini, Peter L. Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004b.

Su-In Lee, Honglak Lee, Pieter Abbeel, and Andrew Y. Ng. Efficient l1 regularized logistic regression. In *The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference*, pages 401–403, 2006.

Yu-Feng Li, Ivor W. Tsang, James T. Kwok, and Zhi-Hua Zhou. Convex and scalable weakly labeled svms. *Journal of Machine Learning Research*, 14(1):2151–2188, 2013.

Weiwei Liu and Ivor W. Tsang. Large margin metric learning for multi-label prediction. In *AAAI*, pages 2800–2806, 2015a.

Weiwei Liu and Ivor W. Tsang. On the optimality of classifier chain for multi-label classification. In *NIPS*, pages 712–720, 2015b.

Weiwei Liu and Ivor W. Tsang. Sparse perceptron decision tree for millions of dimensions. In *AAAI*, pages 1881–1887, 2016.

Xinwang Liu, Lei Wang, Jianping Yin, Yong Dou, and Jian Zhang. Absent multiple kernel learning. In *AAAI*, pages 2807–2813, 2015.

Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevikj, and Saso Dzeroski. An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, 45(9): 3084–3104, Sep 2012.

Qi Mao, Ivor Wai-Hung Tsang, and Shenghua Gao. Objective-guided image annotation. *IEEE Transactions on Image Processing*, 22(4):1585–1597, 2013.

Julian J. McAuley, Rahul Pandey, and Jure Leskovec. Inferring networks of substitutable and complementary products. In *SIGKDD*, pages 785–794, 2015a.

Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes. In *SIGIR*, pages 43–52, 2015b.

Robert J. McQueen, Garner S.R., Nevill-Manning C.G., and Witten I.H. Applying machine learning to agricultural data. *Computers and Electronics in Agriculture*, 12(4):275–293, 1995.

Shahar Mendelson. Rademacher averages and phase transitions in glivenko-cantelli classes. *IEEE Trans. Information Theory*, 48(1):251–263, 2002.

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

Tom M. Mitchell, Rebecca A. Hutchinson, Radu Stefan Niculescu, Francisco Pereira, Xuerui Wang, Marcel Adam Just, and Sharlene D. Newman. Learning to decode cognitive states from brain images. *Machine Learning*, 57(1-2):145–175, 2004.

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

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

Hidekazu Oiwa and Ryohei Fujimaki. Partition-wise linear models. In *NIPS*, pages 3527–3535, 2014.

Jan Paralic and Peter Bednar. Text mining for documents annotation and ontology support, 2003.

E. Y. Pee and Johannes O. Royset. On solving large-scale finite minimax problems using exponential smoothing. *Journal of Optimization Theory and Applications*, 148(2):390–421, 2011.

John C. Platt, Nello Cristianini, and John Shawe-Taylor. Large margin dags for multiclass classification. In *NIPS*, pages 547–553, Denver, Colorado, November 1999. MIT Press.

Yashoteja Prabhu and Manik Varma. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *SIGKDD*, pages 263–272, August 2014.

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

Alain Rakotomamonjy, Francis R. Bach, Stéphane Canu, and Yves Grandvalet. Simplemkl. *Journal of Machine Learning Research*, 9:2491–2521, 2008.

Jesse Read, Bernhard Pfahringer, Geoffrey Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine Learning*, 85(3):333–359, 2011.

Steven Roman. *Coding and Information Theory*. Springer Verlag, New York, 1992.

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

Steven Salzberg, Rupali Chandar, Holland Ford, Sreerama K. Murthy, and Richard L. White. 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, 1995.

Robert E. Schapire and Yoram Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2-3):135–168, 2000.

Claude Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

John Shawe-Taylor and Nello Cristianini. Data-dependent structural risk minimization for perceptron decision trees. In *NIPS*, pages 336–342, 1997.

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

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

Maurice Sion. On general minimax theorems. *Pacific Journal of Mathematics*, 8(1):171–176, 1958.

Yan Song, Xian-Sheng Hua, Li-Rong Dai, and Meng Wang. Semi-automatic video annotation based on active learning with multiple complementary predictors. In *Multimedia Information Retrieval*, pages 97–104, 2005.

Sören Sonnenburg, Gunnar Rätsch, Christin Schäfer, and Bernhard Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006.

Niranjan A. Subrahmanya and Yung C. Shin. Sparse multiple kernel learning for signal processing applications. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(5):788–798, 2010.

Mingkui Tan, Ivor W. Tsang, and Li Wang. Towards ultrahigh dimensional feature selection for big data. *JMLR*, 15(1):1371–1429, 2014.

Jinhui Tang, Xian-Sheng Hua, Guo-Jun Qi, Yan Song, and Xiuqing Wu. Video annotation based on kernel linear neighborhood propagation. *IEEE Transactions on Multimedia*, 10 (4):620–628, 2008.

Grigorios Tsoumakas and Ioannis P. Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. In *ECML*, pages 406–417, Warsaw, Poland, 2007. Springer-Verlag.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Effective and efficient multilabel classification in domains with large number of labels. In *Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08)*, 2008.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis P. Vlahavas. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook*, pages 667–685. Springer US, 2010.

De Wang, Danesh Irani, and Calton Pu. Evolutionary study of web spam: Webb spam corpus 2011 versus webb spam corpus 2006. In *8th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pages 40–49, 2012.

Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus F. M. Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, and Dan Steinberg. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2008.

Zhixiang Eddie Xu, Matt J. Kusner, Kilian Q. Weinberger, and Minmin Chen. Cost-sensitive tree of classifiers. In *ICML*, pages 133–141, 2013.

Zhixiang Eddie Xu, Gao Huang, Kilian Q. Weinberger, and Alice X. Zheng. Gradient boosted feature selection. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 522–531, 2014.

Ian En-Hsu Yen, Xiangru Huang, Pradeep Ravikumar, Kai Zhong, and Inderjit S. Dhillon. Pd-sparse : A primal and dual sparse approach to extreme multiclass and multilabel classification. In *ICML*, pages 3069–3077, 2016.

Yiteng Zhai, Yew-Soon Ong, and Ivor W. Tsang. The emerging "big dimensionality". *IEEE Computational Intelligence Magazine*, 9(3):14–26, 2014.

Cun-Hui Zhang. Nearly unbiased variable selection under minimax concave penalty. *Annals of Statistics*, 38(2):894–942, 2010a.

Cun-Hui Zhang and Jian Huang. The sparsity and bias of the lasso selection in high-dimensional linear regression. *Annals of Statistics*, 36(4):1567–1594, 2008.

Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *TKDE*, 26(8):1819–1837, Aug 2014.

Tong Zhang. Analysis of multi-stage convex relaxation for sparse regularization. *Journal of Machine Learning Research*, 11:1081–1107, 2010b.

Yi Zhang and Jeff G. Schneider. Multi-label output codes using canonical correlation analysis. In *Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 873–882. JMLR.org, 2011.

Yi Zhang and Jeff G. Schneider. Maximum margin output coding. In *ICML*, pages 1575–1582, Edinburgh, Scotland, July 2012. Omnipress.

Ji Zhu, Saharon Rosset, Trevor Hastie, and Robert Tibshirani. 1-norm support vector machines. In *NIPS*, pages 49–56, 2003.

Paul C. Zikopoulos, Dirk deRoos, Krishnan Parasuraman, Thomas Deutsch, David Corrigan, and James Giles. *Harness the Power of Big Data – The IBM Big Data Platform.* Mcgraw-Hill, New York, 2012.