Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Addressing Imbalance in Multi-Label Classification Using Structured Hellinger Forests

Zachary A. Daniels, Dimitris N. Metaxas
Department of Computer Science
Rutgers, The State University of New Jersey
zad7@cs.rutgers.edu, dnm@cs.rutgers.edu

Abstract

The multi-label classification problem involves finding a model that maps a set of input features to more than one output label. Class imbalance is a serious issue in multi-label classification. We introduce an extension of structured forests, a type of random forest used for structured prediction, called *Sparse Oblique Structured Hellinger Forests (SOSHF)*. We explore using structured forests in the general multi-label setting and propose a new imbalance-aware formulation by altering how the splitting functions are learned in two ways. First, we account for cost-sensitivity when converting the multi-label problem to a single-label problem at each node in the tree. Second, we introduce a new objective function for determining oblique splits based on the Hellinger distance, a splitting criterion that has been shown to be robust to class imbalance. We empirically validate our method on a number of benchmarks against standard and state-of-the-art multi-label classification algorithms with improved results.

Multi-label classification is an important unsolved problem in artificial intelligence encountered often in a wide range of domains including tag prediction for images and audio, multiple object recognition in images, predicting gene expression, and classifying documents into predetermined topics. Binary multi-label classification is a special case that involves finding a model that maps a set of input features $X \in \mathbb{R}^{1\times m}$ to more than one binary output label $Y \in \{0,1\}^{1\times n}$.

Class imbalance is a common and challenging problem in multi-label classification. We can view imbalance from two perspectives: imbalance *between* labels (i.e. label 1's positive class appears ten times more frequently than label 2's) and imbalance *within* labels (i.e. label 1 has ten times more positive examples than negative examples). Imbalance is well-studied for single label classification; however, traditional methods for correcting imbalance such as under- and oversampling do not easily generalize to the multi-label domain due to imbalance between labels. Imbalance is typically addressed by using specialized imbalance-aware algorithms or by incorporating cost-sensitivity into existing methods.

We propose a new algorithm for addressing imbalance in binary multi-label classification based on the structured decision forest model (Dollár and Zitnick 2013), a random forest model for structured prediction problems. Structured prediction involves predicting multiple labels by exploiting some structure between labels, e.g. detecting semantic edges (represented as a grid of pixels) in images. Learning multi-label classifiers can be challenging because of the exponentially large label space when considering all possible label combinations. To make learning such classifiers tractable, we assume that labels are correlated with one another: a.) for multiple object recognition, we can exploit co-occurrence relationships between objects, b.) for gene prediction, we can make use of the fact that specific genes often activate (or deactivate) at the same time as other genes, and c.) for document classification, we can utilize topic context, e.g. articles about sports might be more closely related to articles about entertainment than world politics. These correlations represent a special 'structure'. Structured forests effectively learn how to identify and utilize these correlations.

Our model, *Sparse Oblique Structured Hellinger Forests (SOSHF)*, extends structured forests by explicitly addressing imbalanced data. First, we incorporate cost-sensitivity into the clustering step, and second, we learn splits using a criterion based on the Hellinger distance which has been shown experimentally and theoretically to be more robust to class imbalance than standard criteria such as information gain (Cieslak et al. 2012). We found that SOSHF significantly outperforms simple baseline structured forest models as well as state-of-the-art multi-label methods in terms of well-accepted metrics on benchmark datasets across a wide span of domains.

The paper is organized as follows. First, we offer a brief overview of existing multi-label classification algorithms and structured forests. Then, we discuss the technical details of our proposed work. Finally, we empirically evaluate our method against a number of existing methods on benchmark datasets and discuss some of the strengths and weaknesses of our model.

Related Work

Multi-label classification is an area of active research. Tsoumakas et al. divide the set of algorithms for multi-label classification into two sets: those that transform the problem and those that adapt existing algorithms (Tsoumakas, Katakis, and Vlahavas 2009). We first discuss multi-label classification algorithms that transform the problem. Bi-

nary relevance methods are perhaps the simplest transformation. These methods assume independence between labels, and learn disjoint models for each label. "Ranking by pairwise comparison" methods consider every pair of labels (Hüllermeier et al. 2008). Binary classifiers learn how to distinguish which of two labels is present in every pair of disjoint labels, and the output of all the classifiers are combined to predict a full set of labels for a given instance. Classifier chains learn classifiers for single labels sequentially, making use of the predictions of classifiers that exist earlier in the chain (Read et al. 2009). Label powerset methods treat every set of labels present in the training data as its own label, and a single-label classifier is constructed to distinguish between these powersets. There are a number of algorithms that extend the aforementioned label transformations such as (Fürnkranz et al. 2008) and (Tsoumakas and Vlahavas 2007).

Alternatively, one can adapt existing algorithms to the multi-label domain. ML-kNN is an extension of the k-nearest neighbor algorithm (Zhang and Zhou 2007). Clare and King adapt C4.5 decision trees to multi-label classification (Clare and King 2001). BP-MLL modifies the back-propogation algorithm used in neural networks (Zhang and Zhou 2006). Logistic regression is adapted in the IBLR algorithm (Cheng and Hüllermeier 2009). Elisseeff and Weston propose a kernel method for large margin multi-label learning (Elisseeff and Weston 2001). Many other single-label algorithms have been adapted to work with multiple labels. A more extensive overview of the field can be found in several survey papers: (Tsoumakas, Katakis, and Vlahavas 2009), (de Carvalho and Freitas 2009), (Madjarov et al. 2012), and (Zhang and Zhou 2014).

Imbalance in multi-label classification has been a topic of recent interest in the AI community. Some have proposed methods based on extending sampling methods (Charte et al. 2015a) (Charte et al. 2015b) while others have proposed new imbalance-aware algorithms (Zhang, Li, and Liu 2015) (Wu, Lyu, and Ghanem 2016).

Structured Forests

Decision trees (Quinlan 1986) are a popular family of machine learning algorithms learned by recursively partitioning a set of training instances based on some splitting criterion. A function is learned at every node for deciding how to partition the remaining training examples. Once the tree is learned, a new instance can be classified by traversing it using the learned splitting functions, and when a leaf node is reached, the majority label of the training instances that belong to that node is assigned to the new instance. Random forests arise from ensembling multiple decision trees learned from different subsets of training instances and/or features (Breiman 2001). A more complete overview of decision trees and random forests can be found in (Criminisi, Shotton, and Konukoglu 2012).

Structured (decision) forests are random forests used for structured prediction (Dollár and Zitnick 2013). The training and test procedure for structured forests and traditional random forests are identical except for how the splitting functions are learned. Traditional decision trees learn a splitting function at each node that optimizes some criterion based on a set of input features and a single output label. The "correct" form of the objective function to optimize becomes less clear when more than one label exists, especially when the labels exhibit some special structure. Instead of trying to come up with new, specialized splitting criteria to handle these complex problems, Dollár and Zitnick proposed learning a transformation (as simple as k-means clustering) at each node during training that maps the multiple, structured labels to a single label. Following this step, some standard, single label-based splitting criterion is optimized. After the structure and parameters of the tree are learned, the transformations are discarded, and the test stage remains unchanged. Structured trees can be bagged to form forests.

Proposed Work: Sparse Oblique Structured Hellinger Forests (SOSHF)

Sparse Oblique Structured Hellinger Forests (SOSHF) extend structured forests to be able to handle imbalanced data better. Structured forests and SOSHF mostly differ in how the splitting function is learned. Recall that we have two types of label imbalance: between-label and within-label imbalance. We modify the clustering step by incorporating cost-sensitivity to account for both within- and between-label class imbalance. The clustering step transforms the problem into a standard binary single-label problem. Sometimes this step results in clusters of unequal size, so we examine the use of Hellinger distance, which has been shown to be robust to within-label class imbalance in single-label problems, to find better splits (Cieslak et al. 2012).

Step 1: Cost-Sensitive Clustering

Standard structured forests do not have a mechanism in place for dealing with imbalanced data; however, it is easy to extend structured forests to handle class imbalance. We introduce a cost-sensitive clustering step. We need to investigate two sources of imbalance. The first source is *global imbalance* over all training examples. The second source is unique to tree-based classifiers: *local imbalance* over the training examples at a node. Instead of directly clustering on the label-space, we can weigh each label by a corresponding cost determined by it's inverse document frequency at the global and local levels and then perform (weighted) k-means clustering. In our experiments, we use the weighing scheme in Eq. 1 where N_g is the total number of training instances, N_l is the total number of training instances at a node, n_{gi} is the number of positive instances of label i, n_{li} is the number of positive instances of label i present at the node during training, and $\beta \in [0, 1]$ is the mixing coefficient which adjusts the balance between the importance of global and local imbalance. Empirically, we found $\beta = 0.5$ to work well, but in future models, we could attempt to learn this parameter. The weighing scheme helps correct for both the within- and between-label imbalance. It increases the importance of the minority class for each label which helps to reduce the impact of within-label imbalance. It also gives a cost for each label compared to the others that is scale-appropriate, so more importance is given to labels with sig-

nificant imbalance problems during the clustering step, adjusting for between-label imbalance.

$$IDF(i) = \beta * \frac{\log(1+\frac{N_g}{n_{gi}})}{\max_j \log(1+\frac{N_g}{n_{gj}})} + (1-\beta) * \frac{\log(1+\frac{N_L}{n_{li}})}{\max_j \log(1+\frac{N_L}{n_{lj}})} \quad (1)$$

Step 2: The Sparse Hellinger Loss

When using k-means clustering, we often get clusters of different sizes. We've traded the problem of imbalance over the original multi-label space for imbalance over the transformed, single-label space. Luckily, it is conceptually much easier to fix imbalance in this case. We propose using a splitting criterion that is well-suited for imbalanced data. Cieslak et al. showed that standard splitting criteria such as information gain are often ill-suited for imbalanced data and instead proposed using Hellinger distance, a measure of separation between two probability distributions (Cieslak et al. 2012). One way of formulating the Hellinger distance for classification problems is to look at the true positive rate tpr and false positive rate fpr of some set of label assignments. Hellinger distance can be computed as:

$$d_H(tpr, fpr) = \sqrt{(\sqrt{tpr} - \sqrt{fpr})^2 + (\sqrt{1-tpr} - \sqrt{1-fpr})^2} \quad (2)$$

Note that if the fpr is higher than the tpr, we can flip the label assignments, and the distance doesn't change.

It has been empirically observed that Hellinger distance decision trees (HDDT) tend to produce deeper trees because they more finely partition the data (i.e. splits are less balanced). To account for this, we use oblique trees which learn non-axis-parallel but still linear splits of the feature space at each node (Heath, Kasif, and Salzberg 1993). Because oblique trees are more expressive in how they partition the space, they are generally shallower than their axis-parallel counterparts, and we get the additional benefit that they generally perform better.

We need to maximize the Hellinger distance while adjusting a set of weights that produce hard assignments using a linear separating hyperplane. We will use a first-order, gradient-based method to perform the optimization, requiring a differentiable (with respect to the weight vector) form of the Hellinger distance and a differentiable approximation of the hard assignment step. Montillo et al. showed how to make information gain differentiable (Montillo et al. 2013). We construct a similar differentiable loss function that approximately maximizes the squared Hellinger distance. Our approach is similar to performing logistic regression at each node with alternative objective and sigmoid functions.

To simplify the math, we map the labels {0,1} to {-1,1}. Let N represent the number instances used in learning the hyperplane and P represent the number of input variables (features) and a bias term. α is a constant parameter for the sharpness of the sigmoid; $\vec{w} \in R^{P\times 1}$ is the set of weights for the linear hyperplane with the bias term appended; $X \in R^{N\times P}$ is the set of features for all instances used in learning the hyperplane where the last column is all ones; $\vec{y} \in \{-1,1\}^{N\times 1}$ is the true *cluster assignment* for all instances used in learning the hyperplane; and $[y_i = c]$ is an indicator function. We use a sigmoid-shaped function ϕ which will produce a value between 0 and 1 to approximate the hard assignment:

$$\vec{\phi}(\alpha, \vec{w}, X) = \frac{\tanh(\alpha * X\vec{w}) + 1}{2} \quad (3)$$

$$\nabla\vec{\phi}(\alpha, \vec{w}, X) = \frac{\alpha * X^T \operatorname{sech}^2(\alpha * X\vec{w})}{2} \quad (4)$$

As the value of α increases, more importance is placed on correcting cases that are closer to the hyperplane. In our experiments, we start with a small $\alpha(= 0.1)$ and increase it to 1.5 times its current value after every ten iterations of the optimization procedure until $\alpha = 4$.

We can then approximate the values of tpr, fpr, $1-tpr$, and $1-fpr$ where ϕ_i is shorthand for $\phi(\alpha, \vec{w}, \vec{x_i})$:

$$\widetilde{tpr} = \frac{\sum_{i=1}^N([y_i=1]*\phi_i)}{\sum_{i=1}^N[y_i=1]}, \widetilde{fpr} = \frac{\sum_{i=1}^N([y_i=-1]*\phi_i)}{\sum_{i=1}^N[y_i=-1]},$$
$$\widetilde{1-tpr} = \frac{\sum_{i=1}^N([y_i=1]*(1-\phi_i))}{\sum_{i=1}^N[y_i=1]}, \widetilde{1-fpr} = \frac{\sum_{i=1}^N([y_i=-1]*(1-\phi_i))}{\sum_{i=1}^N[y_i=-1]} \quad (5)$$

Plugging these values into Equation 2, we get a differentiable approximation of the Hellinger distance. We call the negative of the square of this differentiable Hellinger distance, the Hellinger loss:

$$L(\alpha, \vec{w}, X, \vec{y}) = -\tilde{d}_H^2(\alpha, \vec{w}, X, \vec{y}) \quad (6)$$

$$\begin{aligned}\nabla L(\alpha, \vec{w}, X, \vec{y}) = &(\sqrt{\widetilde{fpr}} - \sqrt{\widetilde{tpr}}) \\ &* \left(\frac{\nabla\vec{\phi}(\alpha,\vec{w},X_{i:y_i=1})}{(\sum_{i=1}^N[y_i=1])*\sqrt{\widetilde{tpr}}} - \frac{\nabla\vec{\phi}(\alpha,\vec{w},X_{i:y_i=-1})}{(\sum_{i=1}^N[y_i=-1])*\sqrt{\widetilde{fpr}}}\right) \\ &+(\sqrt{\widetilde{1-fpr}} - \sqrt{\widetilde{1-tpr}}) \\ &* \left(-\frac{\nabla\vec{\phi}(\alpha,\vec{w},X_{i:y_i=1})}{(\sum_{i=1}^N[y_i=1])*\sqrt{\widetilde{1-tpr}}} + \frac{\nabla\vec{\phi}(\alpha,\vec{w},X_{i:y_i=-1})}{(\sum_{i=1}^N[y_i=-1])*\sqrt{\widetilde{1-fpr}}}\right)\end{aligned} \quad (7)$$

Recall that we're learning oblique decision trees where each split is parameterized by a set of weights with potentially large dimensionality. As such, oblique decision trees can be prone to overfitting, especially at deeper nodes in the tree where there are few training examples remaining, but the dimension of the feature space might be large. We add a penalty function (regularizer) that encourages the weights to be sparse based on a smooth approximation of the $\ell 1$-norm of the weights (Schmidt, Fung, and Rosales 2007):

$$\Phi(\vec{w}, \eta) = \frac{1}{\eta}[\log(1+\exp(-\eta * \vec{w})) - \log(1+\exp(\eta * \vec{w}))] \quad (8)$$

Higher values of η result in better approximations of the $\ell 1$-norm. We use $\eta = 1000000$.

Our complete optimization problem becomes:

$$\min_{\vec{w}} L(\alpha, \vec{w}, X, \vec{y}) + \gamma * \Phi(\vec{w}, \eta) \quad (9)$$

where γ is a parameter that controls the tradeoff between the loss and penalty functions. This function is entirely differentiable, so standard gradient-based solvers can be used to find a local minimum. Determining a good value of γ is tricky,

especially due to the difference in scale between the loss and penalty functions. In our experiments, we use a heuristic to determine a good scale for gamma. We solve the optimization in two stages. We begin by initializing the weights to be the solution of the least squares problem: $X\vec{w} = \vec{y}$. In the first stage, we ignore the penalty function, minimizing $-\tilde{d}_H^2$ only. This means we've found a separating hyperplane that probably overfits the data, but it also gives us a guess at the scale of the loss near local optimality. Now, we can solve for gamma:

$$\gamma = \rho * \frac{\tilde{d}_H^2(\alpha, \vec{w}, X, \vec{y}_{clus})}{||\vec{w}||_1} \quad (10)$$

ρ is parameter that weighs the importance of sparsity against separability, e.g. $\rho = 0.5$ means finding sparse weights is about half as important as finding a good separating hyperplane. ρ is a much more interpretable parameter than γ and easier to tune. We use the heuristic: $\rho = \min(\log(1 + \frac{|features|}{|instances\_node|}), 0.4)$ which adjusts sparsity by considering the ratio between the number of features and training instances at a node. We then solve the complete optimization problem using γ.

To solve the optimization, we use ADAM with a learning rate of 0.05 and the suggested parameters (Kingma and Ba 2015). ADAM is a stochastic optimization method that adaptively adjusts the learning rate, usually leading to fast, stable convergence to a good local minimum. For large problems, if the number of training instances at a node is large, we can process it in mini-batches without suffering too much of a performance hit. We use batch sizes of min(1000,number of training instances at node), and the data is randomly shuffled every time we pass through the entire set of training instances at a node.

At test time, given a new instance $\vec{x}$, we traverse the tree by checking $\vec{x}\vec{w} \geq 0$ at each node. If no child exists, we recover the predicted labels at the last reachable node $node.\hat{\vec{y}}$.

Ensembling Procedure and Label Assignment

In our experiments, we incorporate three sources of randomness. For each tree we randomly sample 75% of the features, training instances, and labels without replacement. We use the following rule: if a label is used in learning a tree, we weigh that tree's prediction for that label five times higher than for trees where the label is not used in the training procedure. Every tree predicts all labels simultaneously. We use the forest to estimate the likelihood that an instance belongs to the positive class for each label. We select thresholds for making hard assignments by maximizing the f-measure of the predictions of the out-of-bag training instances for each label.

Experiments and Results

We conduct experiments over ten datasets drawn from a wide range of domains: CAL500 (Turnbull et al. 2008), Emotions (Trohidis et al. 2008), Medical (Pestian et al. 2007), the Enron Corpus (Klimt and Yang 2004), Scenes (Boutell et al. 2004), Yeast (Elisseeff and Weston 2001), Corel5k (Duygulu et al. 2002), RCV1 Subsets 1 and 2 (Lewis et al. 2004), TMC2007 (Srivastava and Zane-Ulman 2005), and Mediamill (Snoek et al. 2006). All datasets are provided by the MULAN project (Tsoumakas et al. 2011). We use the pre-extracted features from each dataset. Statistics about the datasets appear in Table 1. Definitions for the label density (LD) and proportion of distinct labels (PDL) can be found in (Tsoumakas, Katakis, and Vlahavas 2009).

We compare with fourteen algorithms. We follow an experimental procedure similar to (Zhang, Li, and Liu 2015). For each dataset, we evenly and randomly split the dataset into training and test sets. We repeat this process eight times per dataset to ensure our results are not strongly influenced by a single split. For datasets with numeric features, we standardize the columns by subtracting the mean and dividing by the standard deviation of the training instances. We remove nominal features which occur in less than one percent of the data. We also remove rare features from the RCV1 dataset by only keeping features that are non-zero in more than one percent of the data. We remove labels with an imbalance ratio ($IR = \frac{|majority\ class|}{|minority\ class|}$) greater than 50 as in (Zhang, Li, and Liu 2015).

We use the the macro-f-measure and macro-AUC for evaluation. These involve averaging the f-measure and AUC values over all labels. For both measures, we perform two tests of significance. First, we consider the methodology of Demsăr: applying the Friedman test followed by the mean-ranks posthoc test with Bonferroni correction (Demšar 2006). Benavoli recently raised some concerns with this methodology: significance can vary depending on the subset of algorithms selected for comparison purposes (Benavoli, Corani, and Mangili 2015). We also follow and report the results of Benavoli's suggested methodology: applying the Friedman test followed by the Wilcoxon signed rank test with Bonferroni correction. For both tests, we use a p-value of 0.05 and also compute the mean rank across all datasets.

Algorithms for Comparison

Binary Relevance (BR) Methods We experiment with two popular models for classification: linear support vector machines (SVM) and random forest (RF) with a cross entropy splitting criterion. We correct for imbalance by undersampling the majority class, oversampling the minority class using ADASYN (He et al. 2008) (an extension of SMOTE (Chawla et al. 2002)), and using imbalance-aware learning. To perform imbalance-aware learning with SVMs, observation weights are penalized according to the probability of seeing the observation's class. For random forests, we adjust the threshold for hard assignment from the predicted class probabilities based on maximizing the f-measure of the out-of-bag instances. When using random forest, we train a single forest for each label.

Higher-Order Methods and Structured Decision Forests We compare against a number of higher-order multi-label classifiers. We use the MULAN library implementations of all algorithms with default parameters unless otherwise noted and C4.5 decision trees as base learners (Tsoumakas et al. 2011). We compare with Multi-Label k-Nearest Neighbor

Dataset	Domain	|Instances|	|Features|	|Labels|	Type	LD	PDL	Min IR	Max IR	Mean IR
CAL500	Audio	502	68	124	Num	0.20	1.00	1.04	24.10	8.45
Emotions	Audio	593	72	6	Num	0.31	0.05	1.25	3.01	2.32
Medical	BioNLP	978	237	14	Nom	0.08	0.04	2.68	43.45	19.94
Enron	Text	1702	999	24	Nom	0.13	0.32	1.01	43.79	16.15
Scene	Images	2407	294	6	Num	0.18	0.01	3.52	5.61	4.66
Yeast	Bioinfo	2417	103	13	Num	0.32	0.08	1.33	12.58	4.25
Corel5k	Text	5000	499	44	Nom	0.05	0.21	3.46	49.00	29.40
RCV1: Subset1	Text	6000	1475	43	Num	0.06	0.10	3.34	49.42	25.53
RCV1: Subset2	Text	6000	1456	39	Num	0.06	0.08	3.22	47.78	26.37
TMC2007	Text/Sci	28596	278	15	Nom	0.14	0.02	1.45	34.26	13.58
Mediamill	Video	43907	120	29	Num	0.14	0.08	1.75	44.74	16.44

Table 1: Characteristics of the datasets used in our experiments. (Nom = Nominal, Num = Numeric)

(ML-KNN) (Zhang and Zhou 2007), Instance-Based Learning by Logistic Regression (IBLR), Ensembles of Classifier Chains (ECC) (Read et al. 2009) with an ensemble size of 50, Calibrated Label Ranking (CLR) (Fürnkranz et al. 2008), Random k-Labelsets (RAkEL) (Tsoumakas and Vlahavas 2007), Hierarchy Of Multilabel classifiERs (HOMER) (Tsoumakas, Katakis, and Vlahavas 2009) using balanced clustering with four clusters, and CrOss-COupling Aggregation (COCOA) (Zhang, Li, and Liu 2015) with 10 couplings and 50 models per ensemble (except for experiments involving TMC2007 and Mediamill where we use 10 models per ensemble for computational feasibility).

We also compare with baseline structured forests. (SF) is a structured forest without oblique splits, no IDF-weighing before performing k-means clustering, and information gain is used as the splitting criteria. (SF-LR) learns splits using logistic regression. (SF-LR-CS) adds cost sensitive (IDF-weighing) clustering. (SF-H) uses the sparse (oblique) Hellinger loss introduced in this paper. (SF-H-CS) uses the sparse Hellinger loss with cost-sensitive clustering. We try to remain as consistent as possible across all forest-based models. We always require tree leaves to have at least three training examples. When building a tree, we always sample 75% of the instances and features without replacement. We perform no pruning. We always use 50 trees in an ensemble (for $50 \times numLabels$ total trees for the binary relevance methods and 50 trees for the structured forests).

Results and Analysis

We present the results of our experiments in Table 2. Structured forests using the sparse Hellinger loss significantly outperform all of the tested algorithms besides COCOA and binary relevance forests in both the macro-f-measure and macro-AUC. SOSHF have the highest mean rank in terms of both macro-f-measure and macro-AUC. Our results suggest our method works well in general and is competitive with state-of-the-art models and binary relevance forests composed of a significantly larger number of trees.

We see some interesting trends when examining how structured forests behave with different modifications. Our base model is a standard structured forest with univariate splits based on information gain. We start by examining oblique structured forests with splits learned using logistic regression. In some cases, the macro-AUC and macro-f-measure increase notably (e.g. Med, Yeast, TMC, and Media). In other cases, performance decreases. One explanation for this mixed behavior is that it is much easier to overfit oblique trees when there are not enough initial training points or too many features. Also, we have not yet accounted for imbalance, and logistic regression might overly favor the majority class in some cases. When we add cost-sensitivity to the clustering stage, we see SF-LR-CS outperforms SF-LR in seven of the datasets for both metrics but still underperforms the baseline in about half the datasets. This suggests that cost-sensitive clustering might be helpful for oblique structured forests. It also suggests that logistic regression might not be a good splitting function in this setting. Next, we consider using the sparse Hellinger loss. We see SF-H substantially outperforms SF, SF-LR, and SF-LR-CS in the majority of cases. This suggests that the Hellinger loss might be more applicable for our problem and that sparse regularization plays an important role in preventing overfitting. Adding cost-sensitive clustering to SF-H, performance improves in eight cases for f-measure and five cases for AUC, occasionally substantially. When performance worsens, it usually isn't by a significant amount. This suggests cost sensitive clustering might be helpful when using the sparse Hellinger loss, but isn't as useful as when its used with more imbalance-sensitive splitting criteria such as logistic regression. This is slightly surprising given that cost-sensitive clustering and the Hellinger loss were designed to overcome different types of imbalance.

Computational Complexity and Scalability

Some concerns may arise about the computational expense of learning SOSHF. A pessimistic bound on the computational complexity of training a SOSHF is $O(i_1 l n^2 t + m^2 n^2 t + i_2 bmnt)$ where n = (number of training instances), m = (number of features), l = (number of labels), t = (number of trees), b = (batch size for the minimizing the Hellinger loss), i_1 = (maximum number of iterations of Lloyd's algorithm), and i_2 = (maximum number of iterations to perform for the optimization procedure). $O(i_1 l n^2 t)$ relates to Lloyd's algorithm for k-means clustering, $O(m^2 n^2 t)$ to the least squares problem, and $O(i_2 bmnt)$ to the minimization problem.

The asymptotic runtime of training a SOSHF is a bit misleading. We have fast, highly optimized libraries for approximating Lloyd's algorithm and computing the least squares solution. The most expensive components can be solved iter-

	CAL	Emot	Med	Enron	Scene	Yeast	Corel	RCV1	RCV2	TMC*	Media*	Summary
svm-cost	0.261	0.632	0.765	0.354	0.633	0.473	0.216	0.356	0.345	0.584	0.352	(8.5)●▲
svm-down	0.275	0.602	0.617	0.265	0.569	0.460	0.122	0.350	0.313	0.514	0.333	(12.5)●▲
svm-adasyn	0.259	0.618	0.755	0.347	0.630	0.466	0.208	0.355	0.343	0.563	0.343	(9.9)●▲
rf-cost	0.306	0.656	0.795	**0.411**	0.754	0.520	0.211	0.449	0.432	0.702	0.484	(3.2)○ △
rf-down	0.285	0.650	0.705	0.264	0.628	0.484	0.129	0.323	0.295	0.547	0.337	(11.5)●▲
rf-adasyn	0.226	0.651	**0.801**	0.343	0.740	0.479	0.081	0.329	0.305	0.648	0.478	(7.9)○▲
ml-knn	0.073	0.592	0.507	0.152	0.722	0.386	0.030	0.118	0.105	0.483	0.244	(16.9)●▲
iblr	0.228	0.629	0.558	0.219	0.728	0.408	0.055	0.195	0.198	0.504	0.276	(14.5)●▲
ecc	0.094	0.633	0.781	0.296	0.729	0.402	0.051	0.244	0.230	0.617	0.247	(12.5)●▲
clr	0.083	0.593	0.768	0.290	0.633	0.408	0.048	0.233	0.233	0.610	0.265	(14.2)●▲
rakel	0.191	0.613	0.766	0.307	0.692	0.428	0.087	0.309	0.298	0.623	0.374	(11.4)●▲
homer	0.254	0.575	0.764	0.332	0.595	0.443	0.146	0.317	0.305	0.589	0.320	(12.1)●▲
cocoa	0.228	0.660	0.777	0.389	0.743	0.479	0.200	0.390	0.376	0.656	0.454	(5.6)○▲
sf	**0.308**	0.660	0.454	0.324	0.644	0.481	0.202	0.364	0.323	0.587	0.343	(8.3)●▲
sf-lr	0.301	0.659	0.685	0.252	0.690	0.501	0.172	0.260	0.242	0.704	0.445	(9.4)●▲
sf-lr-cs	0.301	0.653	0.727	0.247	0.694	0.505	0.198	0.319	0.301	0.719	0.405	(8.5)●▲
sf-h	0.305	0.682	0.775	0.375	**0.763**	0.521	0.231	**0.476**	**0.453**	0.727	0.496	(2.5)○ △
sf-h-cs	0.306	**0.684**	0.792	0.379	0.758	**0.524**	**0.246**	0.472	0.452	**0.731**	**0.505**	(1.8)○ △

	CAL	Emot	Med	Enron	Scene	Yeast	Corel	RCV1	RCV2	TMC*	Media*	Summary
svm-cost	0.534	0.800	0.962	0.726	0.872	0.659	0.707	0.838	0.835	0.914	0.817	(10.5)●▲
svm-down	0.533	0.774	0.967	0.705	0.864	0.642	0.689	0.897	0.889	0.905	0.798	(11.4)●▲
svm-adasyn	0.532	0.793	0.959	0.721	0.868	0.653	0.700	0.837	0.834	0.907	0.813	(12.2)●▲
rf-cost	0.552	0.838	0.963	0.779	0.939	0.703	0.728	0.899	0.892	0.932	0.835	(5.1)○▲
rf-down	0.552	0.824	0.966	0.700	0.911	0.693	0.673	0.890	0.878	0.924	0.814	(9.0)●▲
rf-adasyn	0.541	0.832	0.962	0.767	0.933	0.693	0.725	0.894	0.885	0.928	0.838	(7.2)○▲
ml-knn	0.515	0.812	0.913	0.654	0.926	0.684	0.587	0.664	0.671	0.855	0.767	(14.8)●▲
iblr	0.508	0.833	0.921	0.686	0.935	0.698	0.650	0.789	0.792	0.881	0.801	(12.4)●▲
ecc	0.557	0.843	0.938	0.736	0.943	0.706	0.604	0.870	0.861	0.882	0.802	(8.5)●▲
clr	0.562	0.794	0.965	0.759	0.896	0.651	0.739	0.898	0.891	0.905	0.805	(8.4)●▲
rakel	0.529	0.798	0.900	0.679	0.894	0.640	0.550	0.738	0.726	0.850	0.736	(16.0)●▲
homer	0.515	0.706	0.930	0.645	0.811	0.593	0.590	0.707	0.707	0.800	0.641	(17.0)●▲
cocoa	0.560	0.839	0.969	**0.787**	0.941	**0.717**	0.732	0.911	0.905	0.928	0.842	(3.4)○ △
sf	0.559	0.835	0.916	0.736	0.901	0.643	0.741	0.880	0.860	0.901	0.755	(10.2)●▲
sf-lr	0.549	0.835	0.960	0.655	0.909	0.683	0.685	0.797	0.785	0.938	0.828	(10.5)●▲
sf-lr-cs	0.556	0.830	0.961	0.643	0.909	0.684	0.704	0.833	0.815	0.942	0.789	(10.6)●▲
sf-h	0.569	0.848	0.976	0.771	0.943	0.708	0.736	**0.922**	**0.912**	0.945	**0.857**	(2.2)○ △
sf-h-cs	**0.572**	**0.848**	**0.977**	0.776	**0.943**	0.706	**0.746**	0.921	0.910	**0.946**	0.855	(1.6)○ △

Table 2: The top table reports the macro-f-measure of classifiers on benchmark datasets. The bottom table reports the macro-AUC of classifiers on benchmark datasets. Parentheses denote the mean rank. ● denotes SF-H-CS is statistically superior at $p = 0.05$ and ○ denotes no significant difference according to the Friedman test with the mean-ranks posthoc test with Bonferroni correction. ▲ denotes SF-H-CS is statistically superior at $p = 0.05$ and △ denotes no significant difference according to the Friedman test with the Wilcoxon signed-rank posthoc test with Bonferroni correction. *COCOA run with ensemble size of 10 due to computational limitations

atively using online or mini-batch algorithms making them more efficient to compute and scalable to large problems. Oblique trees tend to partition the space efficiently, significantly reducing tree depth. At every split, we operate on fewer training samples, so learning splits becomes faster as depth increases. If we're willing to trade a small amount of predictive power, we can achieve better scalability by reducing batchsizes, limiting the maximum number of iterations, and/or adjusting learning rates. We can also learn the cluster centers and initial weights using small samples drawn from the training instances at each node or construct forests based on smaller sampling rates (e.g. use 65% of the training data/features/labels for each tree instead 75%). Finally, SOSHF are trivially easy to parallelize.

On smaller datasets, the SOSHF model tends to be slower than the other methods tested, but trains within a reasonable amount of time. On large datasets and datasets with a large number of labels, we've observed that training SOSHF can be faster than training some of the more complex models like COCOA; however, this is likely partially due to differences in implementation (e.g. programming language, parallelization, etc.). In general, we see a trend of trading improved performance for increased computational cost.

Acknowledgements

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1433187.

References

Benavoli, A.; Corani, G.; and Mangili, F. 2015. Should we really use post-hoc tests based on mean-ranks. *JMLR*.

Boutell, M. R.; Luo, J.; Shen, X.; and Brown, C. M. 2004. Learning multi-label scene classification. *Pattern Recognition* 37(9):1757–1771.

Breiman, L. 2001. Random forests. *Machine Learning* 45(1):5–32.

Charte, F.; Rivera, A. J.; del Jesus, M. J.; and Herrera, F. 2015a. Addressing imbalance in multilabel classification: Measures and random resampling algorithms. *Neurocomputing* 163:3–16.

Charte, F.; Rivera, A. J.; del Jesus, M. J.; and Herrera, F. 2015b. Mlsmote: approaching imbalanced multilabel learning through synthetic instance generation. *KBS* 89:385–397.

Chawla, N. V.; Bowyer, K. W.; Hall, L. O.; and Kegelmeyer, W. P. 2002. Smote: synthetic minority over-sampling technique. *JAIR* 16:321–357.

Cheng, W., and Hüllermeier, E. 2009. Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning* 76(2-3):211–225.

Cieslak, D. A.; Hoens, T. R.; Chawla, N. V.; and Kegelmeyer, W. P. 2012. Hellinger distance decision trees are robust and skew-insensitive. *Data Mining and Knowledge Discovery* 24(1):136–158.

Clare, A., and King, R. D. 2001. Knowledge discovery in multi-label phenotype data. In *PKDD*, 42–53. Springer.

Criminisi, A.; Shotton, J.; and Konukoglu, E. 2012. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends in Computer Graphics and Vision* 7(2–3):81–227.

de Carvalho, A. C., and Freitas, A. A. 2009. A tutorial on multi-label classification techniques. In *Foundations of Computational Intelligence Volume 5*. Springer. 177–195.

Demšar, J. 2006. Statistical comparisons of classifiers over multiple data sets. *JMLR* 7(Jan):1–30.

Dollár, P., and Zitnick, C. L. 2013. Structured forests for fast edge detection. In *ICCV*, 1841–1848.

Duygulu, P.; Barnard, K.; de Freitas, J. F.; and Forsyth, D. A. 2002. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *ECCV*, 97–112. Springer.

Elisseeff, A., and Weston, J. 2001. A kernel method for multi-labelled classification. In *NIPS*, 681–687.

Fürnkranz, J.; Hüllermeier, E.; Mencía, E. L.; and Brinker, K. 2008. Multilabel classification via calibrated label ranking. *Machine Learning* 73(2):133–153.

He, H.; Bai, Y.; Garcia, E. A.; and Li, S. 2008. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *IJCNN*, 1322–1328. IEEE.

Heath, D.; Kasif, S.; and Salzberg, S. 1993. Induction of oblique decision trees. In *IJCAI*, 1002–1007.

Hüllermeier, E.; Fürnkranz, J.; Cheng, W.; and Brinker, K. 2008. Label ranking by learning pairwise preferences. *Artificial Intelligence* 172(16):1897–1916.

Kingma, D., and Ba, J. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Klimt, B., and Yang, Y. 2004. Introducing the enron corpus. In *CEAS*.

Lewis, D. D.; Yang, Y.; Rose, T. G.; and Li, F. 2004. Rcv1: A new benchmark collection for text categorization research. *JMLR* 5(Apr):361–397.

Madjarov, G.; Kocev, D.; Gjorgjevikj, D.; and Džeroski, S. 2012. An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition* 45(9):3084–3104.

Montillo, A.; Tu, J.; Shotton, J.; Winn, J.; Iglesias, J.; Metaxas, D.; and Criminisi, A. 2013. Entanglement and differentiable information gain maximization. In *Decision Forests for Computer Vision and Medical Image Analysis*. Springer. 273–293.

Pestian, J. P.; Brew, C.; Matykiewicz, P.; Hovermale, D. J.; Johnson, N.; Cohen, K. B.; and Duch, W. 2007. A shared task involving multi-label classification of clinical free text. In *Workshop on BioNLP*, 97–104. Association for Computational Linguistics.

Quinlan, J. R. 1986. Induction of decision trees. *Machine Learning* 1(1):81–106.

Read, J.; Pfahringer, B.; Holmes, G.; and Frank, E. 2009. Classifier chains for multi-label classification. In *ECML-PKDD*, 254–269. Springer.

Schmidt, M.; Fung, G.; and Rosales, R. 2007. Fast optimization methods for l1 regularization: A comparative study and two new approaches. In *ECML*, 286–297. Springer.

Snoek, C. G.; Worring, M.; Van Gemert, J. C.; Geusebroek, J.-M.; and Smeulders, A. W. 2006. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *ACMMM*, 421–430. ACM.

Srivastava, A., and Zane-Ulman, B. 2005. Discovering recurring anomalies in text reports regarding complex space systems. In *IEEE Aerospace Conference*, 37.

Trohidis, K.; Tsoumakas, G.; Kalliris, G.; and Vlahavas, I. P. 2008. Multi-label classification of music into emotions. In *ISMIR*, volume 8, 325–330.

Tsoumakas, G., and Vlahavas, I. 2007. Random k-labelsets: An ensemble method for multilabel classification. In *ECML*, 406–417. Springer.

Tsoumakas, G.; Spyromitros-Xioufis, E.; Vilcek, J.; and Vlahavas, I. 2011. Mulan: A java library for multi-label learning. *JMLR* 12(Jul):2411–2414.

Tsoumakas, G.; Katakis, I.; and Vlahavas, I. 2009. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook*. Springer. 667–685.

Turnbull, D.; Barrington, L.; Torres, D.; and Lanckriet, G. 2008. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech, and Language Processing* 16(2):467–476.

Wu, B.; Lyu, S.; and Ghanem, B. 2016. Constrained submodular minimization for missing labels and class imbalance in multi-label learning. AAAI.

Zhang, M.-L., and Zhou, Z.-H. 2006. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE TKDE* 18(10):1338–1351.

Zhang, M.-L., and Zhou, Z.-H. 2007. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition* 40(7):2038–2048.

Zhang, M.-L., and Zhou, Z.-H. 2014. A review on multi-label learning algorithms. *IEEE TKDE* 26(8):1819–1837.

Zhang, M.-L.; Li, Y.-K.; and Liu, X.-Y. 2015. Towards class-imbalance aware multi-label learning. In *IJCAI*, 4041–4047.