Model-Shared Subspace Boosting for Multi-label Classification

Rong Yan
IBM TJ Watson Research
19 Skyline Dr.
Hawthorne, NY, 10523
yanr@us.ibm.com

Jelena Tešić
IBM TJ Watson Research
19 Skyline Dr.
Hawthorne, NY, 10523
jtesic@us.ibm.com

John R. Smith
IBM TJ Watson Research
19 Skyline Dr.
Hawthorne, NY, 10523
jsmith@us.ibm.com

ABSTRACT

Typical approaches to the multi-label classification problem require learning an independent classifier for every label from all the examples and features. This can become a computational bottleneck for sizeable datasets with a large label space. In this paper, we propose an efficient and effective multi-label learning algorithm called model-shared subspace boosting (MSSBoost) as an attempt to reduce the information redundancy in the learning process. This algorithm automatically finds, shares and combines a number of base models across multiple labels, where each model is learned from random feature subspace and bootstrap data samples. The decision functions for each label are jointly estimated and thus a small number of shared subspace models can support the entire label space. Our experimental results on both synthetic data and real multimedia collections have demonstrated that the proposed algorithm can achieve better classification performance than the non-ensemble baseline classifiers with a significant speedup in the learning and prediction processes. It can also use a smaller number of base models to achieve the same classification performance as its non-model-shared counterpart.

Categories and Subject Descriptors

H.1.0 [**Information System**]: Models and Principles

General Terms

Algorithms, Performance, Theory

Keywords

Multi-label classification, random subspace methods, boosting, model sharing

1. INTRODUCTION

Multi-label classification is a concept in learning systems that refers to the simultaneous categorization of a given input example into a set of multiple labels. The need for multi-label classification in large-scale systems is increasing in diverse applications such as music categorization, image and video annotation, text categorization and medical applications. Detecting multiple semantic labels from images based on their low-level visual features [2, 15, 14] and automatically classifying text documents into a number of topics based on their textual context [25, 18] are some of the typical multi-label applications. In reality, multimedia and text data collections can contain hundred thousands to millions of data items, and these items can be associated with thousands of different labels. Most existing algorithms do not scale well to such a high computational demand, because they need to learn an independent classifier for every possible label using all the data samples and the entire feature space. Thus it is a great challenge and opportunity for us to design more advanced multi-label learning algorithms that can learn and predict multiple labels in an accurate and efficient way.

To speed up multi-label classification without performance degradation, one approach is to exploit the information redundancy in the learning space. To this end, researchers have proposed several ensemble learning algorithms based on random feature selection and data bootstrapping. The random subspace method (RSM) [12] takes advantage of both feature space bootstrapping and model aggregation, and combines multiple base models learned on a randomly selected subset of the feature space. RSM considerably reduces the size of the feature space in each base model, and the model computation is more efficient than a classifier directly built on the entire feature space. Combining bagging (bootstrap aggregation) and RSM, Breiman has developed a more general algorithm called *random forest* [4]. Random forest aims to aggregate an ensemble of unpruned classification/regression trees using both bootstrapped training examples and random feature selection in the tree induction process. Random forest can be learned more efficiently than the baseline method, and it has empirically demonstrated superiority compared to a single tree classifier. For the multi-label scenario, above algorithms need to learn an independent classifier for every label or assume that the underlying base models can produce multi-label predictions.

However, they ignore an important fact that different labels are not independent of each other, or orthogonal to one another. On the contrary, multiple labels are usually interrelated and connected by their semantic interpretations, and hence exhibit certain co-occurrence patterns in the data collections. For example, in an image collection, the label "car"

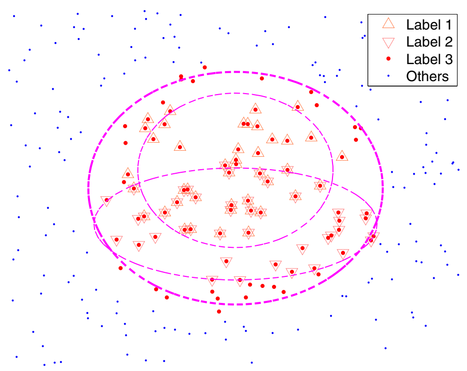

Figure 1: Redundancy in the decision space demonstrated using a 2-D synthetic dataset with 3 labels. The dashed circles indicate the true decision boundary for each label.

almost always co-occurs with the label "road", while the concept "office" is not likely to appear with "road". Inter-related multi-labels exhibit the redundancy of information in the label space. This observation inspires us to further improve the classification efficiency by exploiting the label space redundancy. To achieve this goal in an ensemble learning algorithm, we can allow the underlying base models to be shared across multiple labels and thus the decisions of base models can be re-used in several places at the same time. To illustrate, Figure 1 shows a 2-D synthetic dataset with 3 labels, where its characteristics are explained in Section 6.1. Let us assume that each base model is learned from a depth-1 decision tree that can only produce an axis-parallel decision boundary. In this case, if these models are not allowed to be shared, we need at least 12 base models to construct accurate classifiers for all three labels, because each label requires at least 4 models for capturing its boundaries. However, as we can observe, the decision boundaries of all three labels are closely related to each other. As it results, only 5 base models are needed to construct classifiers of a similar quality, if these models can be shared across multiple decision functions.

In this paper, we propose a boosting-type learning algorithm called model-shared subspace boosting (MSSBoost). It merges ensemble learning, random subspace sampling, and models sharing techniques to automatically find, share and combine a number of random subspace models across multiple labels. MSSBoost is able to reduce the information redundancy in the label space by jointly optimizing the loss functions over all possible labels. Meanwhile, since its base models are built on a small number of bootstrap data samples and a randomly selected feature subspace, the proposed algorithm can scale to sizeable datasets with a large number of labels. Our experiments show that MSSBoost can outperform the non-ensemble baseline classifiers with a significant speedup on the learning and prediction process. It can also use a smaller number of base models to achieve the same classification performance as its non-model-shared counterpart. As a by-product, the proposed algorithm can automatically discover the underlying context between multiple labels, and this offers us more insights to analyze the data collections.

Our following discussions are organized as follows. Section 2 presents the basic notations and terminologies used in this paper. Section 3 illustrates the random space bagging methods in detail. Section 4 describes the proposed MSSBoost algorithm and analyzes its learning properties as well as its computational complexity. Section 5 reviews the related efforts and compares them with our approaches. Section 6 presents the experimental results on multiple data collections and Section 7 concludes our discussions.

2. PRELIMINARIES

In this section, we present the formal notations and terminologies that we use in our study. Let $\mathcal{X}$ be the domain of all possible features, and the number of features for $\mathcal{X}$ is M. Let $\mathcal{Y}$ be a finite set of *labels* of size L, $L = |\mathcal{Y}|$. Without loss of generality, we assume the labels are $\{1, ..., L\}$. In the multi-label classification setting, each data point $\mathbf{x} \in \mathcal{X}$ can be assigned multiple labels from $\mathcal{Y}$. For instance, for an image annotation problem where "building", "sky" and "water" are the possible labels, we can associate an image with the labels of both "building" and "sky". Let $y_l \in \{-1, +1\}$ indicate if the data example x belongs to the l^{th} label, $l \in [1, L]$. Therefore, we can represent a labeled example as $(\mathbf{x}, \mathbf{y})$, where $\mathbf{y} = \{y_l\}_{l=1..L}$. Given a set of training data $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1..N}$, the goal of the multi-label learning algorithms is to produce a *decision function* of the form $F : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$. Here we use a simpler representation by assuming each label l has its own decision function $F_l : \mathcal{X} \rightarrow \mathbb{R}$, and the corresponding prediction y_l is determined by $sign(F_l(x))$.

Ensemble learning approaches such bagging [3] and boosting [8] aim to improve the classification performance by combining the outputs from a family of *base models* $\mathcal{H}$, a.k.a. weak learners. Each base model $h \in \mathcal{H}$ is a binary classifier that produces a real-valued predictions [1] $h : \mathcal{X} \rightarrow \mathbb{R}$, and we assume $|h(\cdot)| \leq 1$. The base models can be generated from different types of algorithms such as decision tree and support vector machines. Some classification algorithms also attempt to optimize a loss function $C(y, f)$, i.e. AdaBoost uses the exponential loss $C(y, f) = \exp(yf(\cdot))$, and LogitBoost uses the logistic loss $C(y, f) = \log(1 + \exp(yf(\cdot)))$. Given the base models available, the learning algorithms can find a linear combination of base models to construct a composite decision function, $F(x) = \sum_t \alpha^t h^t(x)$, where α^t is the combination weight.

3. RANDOM SUBSPACE BAGGING

Bagging, boosting and the random subspace method are among the most popular ensemble learning approaches. Bagging approach was proposed by Breiman [3], and it incorporates the approaches of bootstrapping and classifier aggregation. Multiple sets of training examples can be generated by random sampling the original training set with replacement. The decision functions based on multiple samples can be aggregated by majority voting or averaging. Previous theoretical and empirical results have shown that bagging can reduce the variance of the prediction outputs and thus significantly improve the performance of *unstable* classifiers [3]. However, a simple bagging approach can under-perform in the imbalanced set scenarios, because as multi-label classification problem often faces imbalance distribution in the dataset, the bootstrap examples may only contain few or even none of the minority labeled data. As shown by recent

[1]This setting is more general than most previous work [18] which assumes the base models have to produce multi-label outputs $h : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$.

Algorithm 1 The round-robin random subspace bagging (RSBag) algorithm for multi-label classification.

```
Input: training data $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1...N}$, $\mathbf{x}_i \in \mathbb{R}^M$, number of total base models $T$, number of labels $L$, data sampling ratio $r_d(\leq 1)$, feature sampling ratio $r_f(\leq 1)$.

1. For t = 1 to T,
    (a) Choose a label $l = \mu(t)$, where $\mu(\cdot)$ is the remainder over $L$;
    (b) Take a bootstrap sample $X_+^t$ from the positive data $\{\mathbf{x}_i\}$ for $l$ (i.e., $y_{il} = 1$), $|X_+^t| = Nr_d/2$;
    (c) Take a bootstrap sample $X_-^t$ from the negative data $\{\mathbf{x}_i\}$ for $l$ (i.e., $y_{il} = -1$), $|X_-^t| = Nr_d/2$;
    (d) Take a random sample $F^t$ from the feature indices $\{1, ..., M\}$, $|F^t| = Mr_f$;
    (e) Learn a base model $h^t(\mathbf{x})$ using $X_+^t, X_-^t$ and $F^t$.
    (f) $F_l(\mathbf{x}) \leftarrow F_l(\mathbf{x}) + h^t(\mathbf{x})$.
2. Output the classifier $y_l = sign[F_l(\mathbf{x})]$.
```

studies [22, 7], artificially making the class prior equal by down sampling is usually more effective and efficient than simple bagging. This technique is called "asymmetric bagging" or "balanced bagging".

The random subspace method (RSM) [12] is another example of ensemble learning approach that takes advantages of bootstrapping and aggregating classifiers. However, instead of bootstrapping the training samples, RSM modifies the data examples from the other dimension by randomly selecting a subset of the features to learn the base models. By reducing the size of the feature space, RSM can usually be computed more efficiently than the baseline. The learning performance of RSM can benefit from operating on a small number of features, especially when the number of training examples are insufficient. In this case, more stable classifiers can be built with a relatively larger amount of training data with respect to the feature space.

The connections between bagging and RSM has inspired the development of a more general algorithm called random forest [4], which aggregate an ensemble of unpruned classification/regression trees using both bootstrapped training examples and random feature selection in the tree induction process. Random forest has empirically demonstrated to be able to outperform a single tree classifier such as CART and C4.5. It also yields a generalization error comparable with AdaBoost. However, ensemble learning approaches were not limited to tree classifiers. The extended idea of bagging was applied in a video retrieval task [16], and the random forest idea was used in an image retrieval task [22]. In general, we term the algorithmic combination of bagging and random subspace selection as "random subspace bagging" (RSBag) classifiers (a.k.a., Asymmetric Bagging and Random Subspace classifiers in previous work [22]), and define it as,

DEFINITION 1. *A random subspace bagging classifier is a composite classifier combining a collection of base models $\{h(\mathbf{x}, \Theta^t)\}_{t=1}^T$ where $\{\Theta^t\}$ are independent identically distributed random vectors. Each classifier generates a prediction output based on training data and Θ^t.* [2]

To apply the random subspace bagging algorithms for the multi-label classification problem, we present a round-robin random subspace bagging approach in Algorithm 1. This algorithm first selects a label to work with in a round robin fashion. Then we learn a base model based on Nr_d balanced bootstrap samples from the positive and the negative data, together with Mr_f random samples from the feature indices, where r_d is called the *data sampling ratio* and r_f is called the *feature sampling ratio*. Both sampling ratios are determined by the input parameters and typically they are less than 1. At the end we aggregate all the base models for the same label into a composite classifier. This algorithm is similar to a balanced version of random forests [22, 7] with the random vectors Θ^t as the indices of X_+^t, X_-^t and F^t. The only difference is that it can use any binary classifier to construct the base models without being limited to the decision trees. To shed lights on why the random subspace bagging works, we present an upper bound for its generalization error. The importance of the interaction between two ingredients, the model strength s and the model correlation $\bar{\rho}$, for minimizing the upper bound on the classification error is shown in the following theorem:

THEOREM 1. *For each label l in the random subspace bagging algorithms described in Algorithm 1, an upper bound for the generalization error is given by,*

$$E^*(F_l) \leq \bar{\rho}(1 - s_l^2)/s_l^2,$$

where s_l is the strength of base models for the label l, i.e.,

$$s_l = 2E_{\mathbf{x},y_l} P_\Theta(h(\mathbf{x}, \Theta) = y_l) - 1,$$

and ρ is the correlation between any two base models, i.e.,

$$\bar{\rho} = E_{\Theta,\Theta'}\left[\rho_{\mathbf{x}}(h(\mathbf{x}, \Theta), h(\mathbf{x}, \Theta'))\right].$$

PROOF: Immediately follow the proof of Theorem 2.3 in [4] by viewing multi-label classification as a set of binary classification problems on each label l. □

Therefore, classification accuracy can be improved by minimizing the model correlation $\bar{\rho}$ while maintaining the model strength s. Randomly sampling the training examples and feature subspace is useful to diversify the base models, and meanwhile reduce the model size. As long as the quality of each bootstrap models is not significantly deteriorated, random subspace bagging is able to generate a classifier with smaller model size, while produce a comparable performance with the single classifier learned from the entire data space. This claim is confirmed by the experiments in Section 6.

4. MODEL-SHARED SUBSPACE BOOSTING

The base models learned from bootstrap samples and feature subspace provide a foundation for developing scalable ensemble learning algorithms for multi-label classification. However, because the base models are learned independently on each label, they may contain redundant decision information that can be shared across a number of labels. By exploiting this kind of information redundancy, we can further reduce the size of ensemble classifiers and keep the classification performance intact. To automatically discover the base models that can be shared among labels, we switch the

[2] Θ^t can have different semantics: in bagging Θ^t can be a bootstrapped sample from the integers from 1 to the number of data N, and in RSM Θ^t can be a number of independent random integers between 1 to the number of features M.

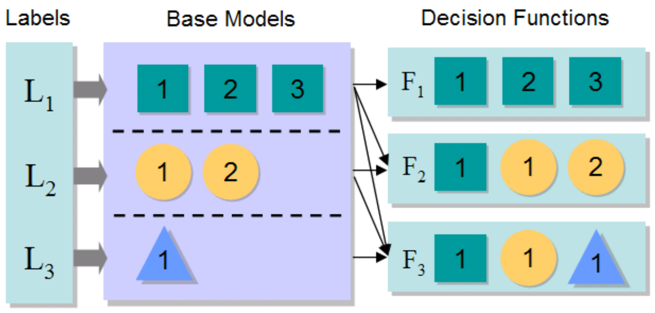

Figure 2: Illustration of sampling, boosting and model sharing of 6 base models for 3 labels.

model aggregation strategies from bagging to a boosting-type algorithm [8], as a boosting-type approach can search the function space for the most useful models in a gradient way. By combining the idea of random subspace sampling, boosting and model sharing, we propose a new algorithm called model-shared subspace boosting (MSSBoost), as illustrated in Figure 2. First, we detail the entire process of the proposed model, followed by an analysis on the learning properties and discussions on the computational requirements so as to emphasize its scalability.

4.1 Model Description

In order to discover the best shared subspace models over the entire label space, MSSBoost iteratively finds the most useful subspace base models, shares them across labels, and combine them into composite classifiers. Algorithm 2 shows the detailed algorithm of MSSBoost. We begin by initializing a pool of K base models $h_k(\cdot)$, where $h_k(\cdot)$ is learned on the label $l(k)$ using random subspace and data bootstrapping methods (i.e., the step 1(b)-(e) in Algorithm 1). In this paper, we simply set $K = L$ and $l(k) = k$, which means only one base model are generated for each label l [3]. In each iteration t, we search the entire model pool for the best fitted model h^t and its corresponding combination weights $\alpha^t = \{\alpha_l^t\}_{l=1}^L$. This can be achieved by minimizing a *joint logistic loss function* summed over all the labels. One important issue here is to select a proper loss function. Extreme loss functions such as the exponential loss functions are not robust against outliers, because they tend to assign much higher weights to the noisy data. Following the idea of LogitBoost [9], we adopt a logistic loss $C(y, f) = \log(1 + e^{-yf})$ and thus the optimization step can be rewritten as,

$$\begin{aligned}\{\alpha^t, h^t\} &= \arg\min_{\alpha^t, h^t} \sum_{il} C(y_{il}, F_l(\mathbf{x}_i) + \alpha_l^t h^t(\mathbf{x}_i)) \\ &= \arg\min_{\alpha_l^t, h_t} \sum_{il} \log\left(1 + e^{-y_{il}(F_l(\mathbf{x}_i) + \alpha_l^t h^t(\mathbf{x}_i))}\right). \quad (1)\end{aligned}$$

To find the optimal h^t and α^t, we can iterate over all candidate models in the pool $\mathcal{H}_p$ and compute the corresponding α_l^t using adaptive Newton steps, of which the details are shown in Section 4.2. After h^t is identified from the pool, we update the decision function $F_l(x)$ for each label using the selected model, and then replace this model by a new model $\bar{h}^t$ learned on the same label as h^t. Finally, the composite classifiers F_l are used to provide the prediction results.

[3]The length of the initial model pool can be larger than the number of labels, but the current settings is sufficient enough to produce reasonable performance for large number of labels.

Algorithm 2 The model-shared subspace boosting (MSSBoost) algorithm for multi-label classification.

Input: training data $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1..N}$, $\mathbf{x}_i \in \mathbb{R}^M$, number of total base models T, number of labels L, data sampling ratio r_d, feature sampling ratio r_f.

```
1. Set $F_l(x) = 0$ for each label $l$. Initialize a pool of candidate models $\mathcal{H}_p = \{h_1(\cdot), ..., h_K(\cdot)\}$, where $h_k(\cdot)$ is learned on the label $l(k)$ using the step 1(b)-(e) in Algorithm 1. We set $K = L$ and $l(k) = k$ in this work.
2. For t = 1 to T,
    (a) Find $\alpha^t = \{\alpha_l^t\}_{l=1..L}$ and $h^t \in \mathcal{H}_p$ that minimize the joint loss function $\sum_{i=1}^N \sum_{l=1}^L C(y_{il}, F_l(\mathbf{x}_i) + \alpha_l^t h^t(\mathbf{x}_i))$ based on Eqn(1);
    (b) $F_l(\mathbf{x}) \leftarrow F_l(\mathbf{x}) + \alpha_l^t h^t(\mathbf{x})$, $\forall l = 1...L$;
    (c) Replace $h^t$ in $\mathcal{H}_p$ with a new candidate model $\bar{h}^t$, which is learned on the same label as $h^t$ using the step 1(b)-(e) in Algorithm 1.
3. Output the classifier $y_l = sign[F_l(\mathbf{x})]$.
```

Remark: Typical boosting algorithms learn base models using a re-weighted distribution in each iteration. Instead of doing this, we decide to select the most useful model from a candidate model pool. We choose this method mainly because of its efficiency. In the traditional boosting strategy, a total number of $L \cdot T$ base models is built in the training process. This is far more expensive than to generate the $L + T$ base models in the current implementation.

It is interesting to show that the random subspace bagging method (RSBag) is a special case of MSSBoost, if we select h^t as the candidate model h_k learned from the label $l(k)$ in a round-robin fashion, and set the corresponding $\alpha_{l(k)}^t = 1$ with other α^t as 0. Moreover, a *non-model-shared* counterpart of MSSBoost, or called non-shared subspace boosting (NSBoost), can be obtained by modifying the step 3(a) in Algorithm 2 as follows: for each candidate model h_k, we only allow the corresponding $\alpha_{l(k)}^t$ to have a non-zero value while setting the other α^t to 0. So in both of these two algorithms, the base models from one label are no longer shared to the F_l of the other labels. We will compare both RSBag and NSBoost with the proposed MSSBoost algorithm in our experiments.

4.2 Computation of the Combination Weights

Given a base model $h^t \in \mathcal{H}_p$, we need to find a set of combination weights α^t that helps to minimize the joint loss function. To this end, we apply an efficient adaptive Newton algorithm. Adaptive Newton algorithm can be summarized as the following weighted least-squares regression step:

$$\alpha_l^t = (H^T W_l H)^{-1} H^T Z_l, \quad (2)$$

where $H = (h^t(\mathbf{x}_1), ..., h^t(\mathbf{x}_N))^T$, $p_{il} = 1/(1 + e^{-F_l(\mathbf{x}_i)})$, $w_{il} = p_{il}(1 - p_{il})$ [4], $W_l = diag(w_{l1}, ..., w_{lN})$ and $z_{il} = y'_{il} - p_{il}$, $y'_{il} = (y_{il} + 1)/2$, $Z_l = (z_{l1}, ..., z_{lN})$.

[4]Sometimes w_i might get very small when p_i is close to 0 or 1, and hence we impose a lower threshold on the weights $w = \max(w, 10^{-6})$.

PROPOSITION 1. *Eqn(2) is a one-step adaptive Newton update for the joint logistic loss function.*

PROOF: Let us consider the update $F_l(\mathbf{x}) + \alpha_l^t h^t(\mathbf{x})$ and the joint loss function with respect to α^t,

$$C(\alpha^t) = \sum_i \sum_l C(y_{il}, F_l(\mathbf{x}_i) + \alpha_l^t h^t(\mathbf{x}_i))$$

We can compute the first derivative and second derivative at $\alpha_l^t = 0, \forall l$,

$$\frac{\partial C(\alpha^t)}{\partial \alpha_l^t} |_{\alpha_l^t=0} = \sum_i \frac{y_{il} h^t(\mathbf{x}_i)}{1 + e^{y_{il} F_l(\mathbf{x}_i)}} = H^T Z_l,$$

$$\frac{\partial C^2(\alpha^t)}{\partial \alpha_l^{t2}} |_{\alpha_l^t=0} = \sum_i \frac{[h^t(\mathbf{x}_i)]^2}{2 + e^{y_{il} F_l(\mathbf{x}_i)} + e^{-y_{il} F_l(\mathbf{x}_i)}} = H^T W_l H.$$

Therefore the Newton update can be written as,

$$\alpha_t^l \leftarrow \left[\frac{\partial C^2(\alpha^t)}{\partial \alpha_l^{t2}}\right]^{-1} \frac{\partial C(\alpha^t)}{\partial \alpha_l^t} = (H^T W_l H)^{-1} H^T Z_l.$$

□

For each model h^t in the candidate model pool, we can compute its corresponding combination weights α^t. Then Eqn(1) is applied to find the best pair of $\{\alpha^t, h^t\}$ that can best minimize the joint loss function. Theoretically, rather than only running one-step update, we can compute multiple consecutive steps of Newton updates in the step 2(a) of Algorithm 2 in order to get a better estimate for α_l^t per iteration. However, since the boosting update also aims to optimize the same criterion in the outer loop, increasing the number of consecutive Newton updates inside will bring little advantage on the final prediction results. Therefore we only run one-step Newton update to compute each α_l^t. This strategy is also adopted in [9].

4.3 Analysis of Learning Properties

In this section, we analyze two learning properties for the proposed algorithm. First, we derive an upper bound for the cumulative training error over all the labels:

THEOREM 2. *Let $F_{il} = F_l(x_i)$. The following upper bound holds for the cumulative training error,*

$$\sum_i \sum_l I(\mathit{sign}(F_{il}) \neq y_{il}) \leq \sum_i \sum_l C(y_{il}, F_{il}),$$

where $I(x) = 1$ if x is true and otherwise $I(x) = 0$.

PROOF: Based on the inequalities $\log(1+e^{-x}) > 0$ and when $x \leq 0$, $\log(1 + e^{-x}) \geq 1$, we can show that

$$\begin{aligned}\sum_i \sum_l I(\mathsf{sign}(F_{il} \neq y_{il}) \quad &\leq \sum_{y_{il}F_{il}\neq 0} \log\left(1 + e^{-y_{il}F_{il}}\right) \\ &\leq \sum_i \sum_l \log\left(1 + e^{-y_{il}F_{il}}\right),\end{aligned}$$

where the right hand side is exactly $\sum_i \sum_l C(y_{il}, F_{il})$ based on its definition. □

Theorem 2 suggests that it is reasonable for the MSS-Boost algorithm to directly optimize the joint logistic loss function, which serves as an upper bound for the training error over all the labels. Therefore minimizing this joint loss function can guarantee to produce a low training error in the learning process. Next, we show the proposed algorithm also satisfy the boosting property [17], i.e., if the hypothesis space of base models satisfy the "weak learnability" (i.e., the weighted margin of base models is at least $\lambda > 0$ in each iteration), the loss function will go towards 0 with sufficiently large training samples, and thus the generalization error will go to 0 with an increasing margin [9]. The satisfaction of the boosting property can be proved as follows,

THEOREM 3. *Suppose the base models h_k in the candidate pool satisfy the weak learnability for the label $l(k)$, i.e.,*

$$\frac{\sum_i w_{il(k)} y_{l(k)} h_k(x_i)}{\sum_i w_{il(k)}} \geq \lambda > 0, \forall k = 1...K \quad (3)$$

where w_{il} is the sampling distribution $\frac{\partial C(y_{il}, F_{il})}{\partial F_{il}} = \frac{1}{1+e^{y_{i1}F_{i1}}}$ for the i^{th} data example. Then for any value S_0, we can show that after at most $T = \frac{2N^4L^4}{\lambda^2 S_0^2}$ rounds we can obtain a joint loss $C(\cdot)$ no more than $S_0 L$.

PROOF SKETCH: Let C_l^t be the total loss for the label l in iteration t, i.e., $C_l^t = \sum_i C(y_{il}, F_{il})$, and the joint loss is $C^t = \sum_l C_l^t$. If the condition $C^t \geq S_0 L$ satisfies, there must be one $C_l^t \geq S_0$. Without loss of generality, let us assume $C_1^t \geq S_0$. In the following we will show the improvement in the joint loss function is at least $\frac{\lambda^2 S_0^2}{2N^3L^3}$ if the candidate model h_1 (learned on the label 1) is selected from the model pool. For the sake of simplicity, we ignore the iteration index t in the rest of the discussion unless stated otherwise.

Because the joint loss function is minimized in each iteration, C_1 must be less than the initial joint loss NL and thus $\log(1 + e^{-y_{i1}F_{i1}}) \leq NL, \forall i$. By combining this fact with the following inequality,

$$-\log \gamma \geq 1 - \gamma \geq \frac{1-\gamma_0}{\log \gamma_0} \log \gamma, \quad \text{if} \quad 0 \leq \gamma_0 \leq \gamma \leq 1,$$

where γ_0 is a constant, we can have

$$\log(1 + e^{-y_{i1}F_{i1}}) \geq \frac{1}{1 + e^{y_{i1}F_{i1}}} \geq \beta_0 \log(1 + e^{-y_{i1}F_{i1}}), \forall i \quad (4)$$

where $\beta_0 = (1 - 2^{-NL})/NL$. Since N and L is large in general, we can simplify $\beta_0 = 1/NL$.

Using the inequalities in (3) and (4), we can derive an lower bound for the negative gradient of C_1 w.r.t. α_1,

$$\begin{aligned}-\sum_i \frac{\partial C(y_{i1}, F_{i1} + \alpha_1 h_{i1})}{\partial \alpha_1} |_{\alpha_1=0} &= \sum_i \frac{1}{1 + e^{y_{i1}F_{i1}}} \cdot y_{i1} h_{i1} \\ \geq \lambda \sum_i \frac{1}{1 + e^{y_{i1}F_{i1}}} &\geq \lambda\beta_0 \sum_i \log(1 + e^{-y_{i1}F_{i1}}) \geq \frac{\lambda S_0}{NL}. \quad (5)\end{aligned}$$

Next, we bound the second derivative of the change in the loss function under the assumption that $|h_{i1}| \leq 1$,

$$\sum_i \frac{\partial^2 C(y_{i1}, F_{i1} + \alpha_1 h_{i1})}{\partial \alpha_1^2} |_{\alpha_1=0} \leq \sum_i \frac{h_{i1}^2}{1 + e^{y_{i1}F_{i1}}} \leq NL. \quad (6)$$

If we combine the two bounds together and take a step size $\alpha_1 = \frac{\lambda S_0}{N^2L^2}$, the reduction of the loss function C_1 is at least $\frac{\lambda^2 S_0^2}{2N^3L^3}$ according to Taylor's theorem. Since the loss function of the other labels will at least be unaffected, the joint loss function C will also at least be reduced by $\frac{\lambda^2 S_0^2}{2N^3L^3}$. In this case, if a different candidate model $h_{k'}$ is selected in MSSBoost, its joint loss reduction must be more than using h_1 based on our model selection criterion. Now since the initial joint loss is NL and thus within this number of iterations, $T = \frac{2N^4L^4}{\lambda^2 S_0^2}$, we must get a non-positive loss if we still have a joint loss large than S_0L. This obviously gives a contradiction and thus the boosting property is proved. □

4.4 Computational Complexity

The computational complexity of MSSBoost mainly comes from (a) the generation of base models from bootstrapped data/feature spaces, and (b) minimization of the joint loss function using adaptive Newton steps. Since the second step only involves the operations of matrix computations and a scalar inverse, its effect on the computational complexity of the overall system is not influential as the calculations are much faster than a generation of base models in first step.

As noted in Section 4.1, our current implementation needs to generate $L+T$ base models in total, where each model is learned with Nr_d examples and Mr_f features. Thus the total time is $(L+T)\cdot t_l(Nr_d, Mr_f)$, where $t_l(n,m)$ is the base model learning time with n examples and m features. Similarly, the total prediction time is $T \cdot t_p(Nr_d, Mr_f)$, where $t_p(n,m)$ is the base model prediction time. The form of t_l and t_p is determined by the choice of the underlying base models. For example, the learning time of decision trees is $t_l(n,m) = O(mn \log n)$ (Chapter 9 in [11]) and the prediction time is $t_p(n,m) = O(mn)$ if we assume the base model size is proportional to the number of data and features. Most current support vector machine(SVM) implementations such as [13] have a learning time of $t_l(n,m) = O(mn^2)$ and a prediction time of $t_p(n,m) = O(mn)$ if the number of support vectors is proportional to the number of training data. Compared with a baseline classifier that learns a model from N examples and M features for each label, MSSBoost can achieve a speedup of $\frac{T+L}{L} r_d r_f$ in learning and $\frac{T}{L} r_d r_f$ in prediction with decision trees. Similarly, with SVMs, MSSBoost can achieve a speedup of $\frac{T+L}{L} r_d^2 r_f$ and $\frac{T}{L} r_d r_f$ in learning and prediction respectively. This can result in a significant speedup in general. For instance, if we set $r_d = 0.1$, $r_f = 0.1$ and $T = 3L$, the learning and prediction time of MSSBoost with SVMs can be reduced around 250 times and 33 times over the baseline.

5. RELATED WORK AND DISCUSSIONS

Ensemble learning approaches such as boosting and bagging [8, 3, 18] have been demonstrated to be effective in improving the classification accuracy of weak learned models. By combining the idea of ensemble learning with random subspace methods, random forest [4] shows the possibility of developing efficient and effective classifiers using a set of tree models. The similar idea has also been applied to the task of image retrieval with SVMs as the base models [22]. Along another line, Caruana et al. [6] proposed an ensemble selection approach for choosing the most useful models from a large model library via a forward stepwise selection method. However, since most ensemble learning approaches are developed on binary classification, their extensions to multi-label classification either need to maintain independent binary classifiers for each label, or require the underlying models to be able to generate multi-label outputs, which prevent them from capturing the common representations across labels in the ensemble aggregation level. In this sense, the AdaBoost.OC algorithm [19] is a more closely related to our work, which merges AdaBoost with the Error-Correcting Output Codes (ECOC) to handle the multi-class learning problem with a pool of binary classifiers. It only needs to keep track of one set of data distributions for multiple classes at the same time. However, because multi-class learning problems assume each data point can only associate one single class, their algorithm is not directly applicable to the multi-label problem.

In the machine learning community, the idea of sharing the common information among multiple labels have been investigated by the methods called "multi-task learning", which has also been known as "transfer learning" and "learning to learn". These methods handle the multi-label classification problem by treating each label as a single task and generalizing the correlations among multiple tasks. From another viewpoint, they can also be thought as learning a common feature representation across all the tasks. Many methods have been proposed for multi-task learning, such as neural networks [5], regularization learning methods [1], hierarchical Bayesian models [26] and so on. For example, Zhang et al. [26] proposed a latent variable model for multi-task learning which assumes that tasks are sparse linear combination of basic classifiers. Ghamrawi et al. [10] proposed a multilabel conditional random field classification model that directly parameterize label co-occurrences in multi-label classification. The listed approaches are usually more computationally expensive than the baseline single-task learning algorithms because they often use the single-task learners in an iterative process and require a complex inference effort to estimate the task parameters. This clearly contrasts with the MSSBoost algorithm which can considerably improve the learning efficiency in multi-label scenarios. In addition, MSSBoost can be build on any base models without being limited to a specific type.

The problem of exploring and leveraging the connections across labels have also been seen in many other research areas. One such example is the domain of image annotation and object recognition which aims to detect a number of scenes and objects in the given images. For instance, Snoek et al. [21] proposed an semantic value chain architecture including a multi-label learning layer called *context link*. Yan et al. [24] studied various multi-label relational learning approaches via a unified probabilistic graphical model representation. However, these methods need to construct an additional computational intensive layer on top of the base models, and they do not provide any mechanisms to reduce the redundancy among labels other than utilizing the multi-label relations. Torralba et al. [23] developed a joint boosting algorithm that finds the common image features shared across multiple labels. But this method are specifically built above the level of image features rather than the general base models in our methods. In addition, its learning process is computationally demanding, as it needs to search either $2^L - 1$ sharing patterns or $L(L+1)/2$ in its simplified version.

6. EXPERIMENTS

We demonstrate the effectiveness and efficiency of the proposed MSSBoost algorithm on several multi-label data collections, including a synthetic dataset and two real-world multimedia collections. The statistics of the data collections are summarized in Table 1.

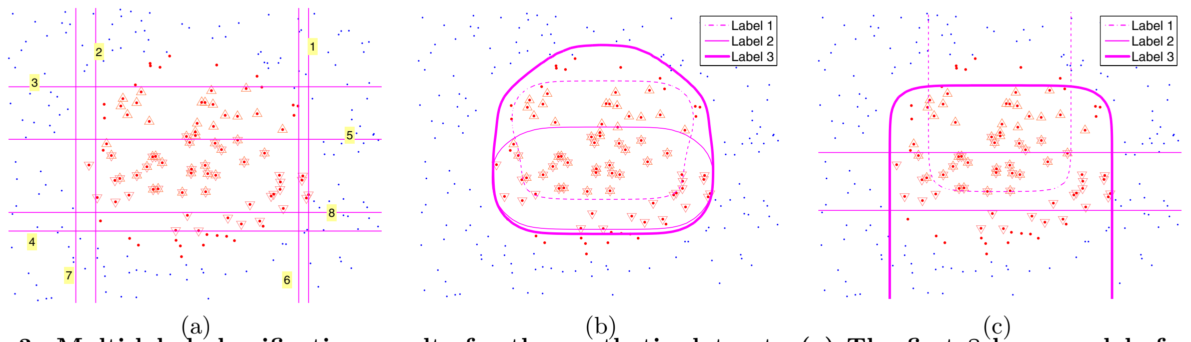

Figure 3: Multi-label classification results for the synthetic dataset. (a) The first 8 base models found by MSSBoost, (b) Decision boundary of MSSBoost with T=8, and (c) Decision boundary of NSBoost with T=8.

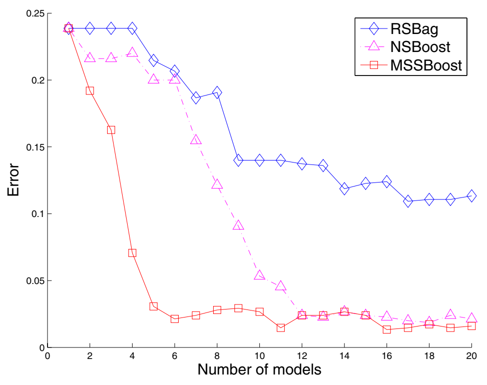

Figure 4: The error curves vs. number of base models used for the synthetic dataset.

Collections	# Data	# Feature	# Label
Synthetic	1000	2	3
TREC	6525	343	39
Consumer	8390	343	33

Table 1: Data collections and their statistics.

6.1 Synthetic Data

To illustrate MSSBoost can automatically discover a set of common base models shared across multiple labels, we generate a synthetic dataset with 3 labels in a two-dimensional feature space. This dataset contains 1000 data points uniformly sampled inside a rectangle area. The samples of each label follow a different Gaussian distribution. In total, we have 200, 200 and 400 samples for the first, second and third label respectively. Some of these samples are associated with multiple labels and some of them do not have any labels. We randomly select 75% of the samples to construct a training set and left the rest to be a testing set. Figure 1 shows the data distribution and their labels in the testing set. The training set has a similar distribution with the testing set. The base models are learned using decision stumps, i.e., decision trees of depth 1, on a randomly selected 1-dimensional feature space and a set of bootstrap training samples with a sample ratio 10%, or equivalently 75 samples.

Figure 3(a) shows the decision boundaries of the first eight base models generated by MSSBoost. Many of these base models can be re-used in the decision functions of several labels. The base model 1 can serve as the right boundary for all three labels, and the base model 2 can serve as the left boundary. With a mere 8 base models but allowing them shared among labels, MSSBoost constructs fairly accurate decision boundaries for all three labels as shown in Figure 3(b). Figure 3(c) shows the decision boundary of NSBoost where the model sharing ability is disabled. We find that the decision boundaries are noticeably worsen even if they use the same number of base models.

To provide a deeper insight for the learning algorithms, Figure 4 plot the learning curves for three learning algorithms, i.e., MSSBoost, NSBoost and RSBag. The number of base models grows from 1 to 20. We can observe that MSSBoost converges to its optimal performance with only 5 base models, while NSBoost requires a number of at least 12 based models to achieve the similar performance and RSBag is not able to capture the most useful models within 20 iterations. This additional decision power of MSSBoost clearly stems from the introduction of model-sharing ability into the learning process.

6.2 Real-world Multimedia Collections

6.2.1 *Experimental Setting*

The following experiments evaluate the proposed algorithms based on a image annotation task on two real-world multimedia collections. The purpose of this task is to categorize a given image or video keyframe into a set of manually defined semantic labels based on low-level visual features.

TREC The TRECVID video collections have become the standard large-scale testbeds for the video annotation and retrieval task over the past couple of years [20]. In particular we use the TRECVID-05 collection, which consists of 220 hours broadcast news videos captured in the fall of 2004 from 11 English, Arabic and Chinese channels. Each video clip is segmented into representative shots and each shot is associated with a keyframe. In this experiment, we compiled a collection of 6525 keyframes randomly sampled from the TRECVID-05 development corpus. The TRECVID-05 participants jointly annotated a development set with 39 semantic labels from LSCOM [14], and made the ground truth publicly available for training purposes.

Consumer The second dataset is a consumer image collection. To model the consumer photo space well, we have asked eight of our colleagues to contribute their personal digital photographs to our collection, which contains a total of 8390 images. Then the same group of people are requested to propose visually relevant labels, and after a couple of iterations we came

	Baseline	Bagging	Subspace	RSBag	NSBoost	MSSBoost
Mean-AP	0.448	0.449±0.003	0.321±0.005	0.416±0.004	0.437±0.002	0.456±0.003
Time(sec.)	5898	41290	27	68	94	94
Model #	39	390	39	100	100	100
Model Size	$\sim 4.3 \times 10^7$	$\sim 3.0 \times 10^8$	$\sim 4.0 \times 10^5$	$\sim 1.3 \times 10^6$	$\sim 1.3 \times 10^6$	$\sim 1.2 \times 10^6$

	Baseline	Bagging	Subspace	RSBag	NSBoost	MSSBoost
Mean-AP	0.332	0.339±0.005	0.267±0.004	0.313±0.004	0.321±0.003	0.341±0.003
Time(sec.)	5158	52369	9	23	31	31
Model #	33	330	33	100	100	100
Model Size	$\sim 3.0 \times 10^7$	$\sim 3.0 \times 10^8$	$\sim 3.0 \times 10^5$	$\sim 8.0 \times 10^5$	$\sim 8.0 \times 10^5$	$\sim 8.0 \times 10^5$

Table 2: Comparison of classification methods including the baseline SVM classifier (Baseline), bagging SVMs (Bagging), subspace SVMs (Subspace), random subspace bagging (RSBag), non-shared subspace boosting (NSBoost) and model-shared subspace boosting (MSSBoost). The learning performance is measured by Mean-AP. The training time is reported in seconds. Baseline/Subspace learn 1 model per label, Bagging learns 10 models per label and the last 3 methods learn 100 models in total. Top: TREC, Bottom: Consumer.

up with 33 labels. We randomly split the collection among these eight colleagues to annotate given images with one or more of 33 labels. The set of labels reflect the fact that the collection contains mostly vacation/holiday photos such as nature and urban scenes.

In both datasets, three types of low-level features are generated for each image [15]: 166-dimensional color correlogram, 81-dimensional color moments, and 96-dimensional co-occurrence texture vector. We concatenate these three features into a 343-dimensional vector and use it as the image representation to learn the visual models. For the task of image annotation, the classification accuracy is not a preferred performance measure since the number of labeled samples is usually much smaller than non-labeled data. Instead, NIST defines average precision as a measure of effectiveness. For each label, let R be the number of true labeled images in the collection of N images. At any given index j, let R_j be the number of correctly labeled images in the top j most confident images predicted by the learning algorithm. Let $I_j = 1$ if the j^{th} image is relevant and 0 otherwise. The average precision (AP) is then defined as $\frac{1}{R}\sum_{j=1}^{N}\frac{R_j}{j}I_j$. Mean average precision (Mean-AP) is the mean of average precisions for all the labels.

We randomly select 75% of the data examples to construct a training set, and use the remaining 25% for testing. The training time are measured on a Windows machine with a 2.16GHz CPU and 2G main memory. The data sampling ratio r_d and the feature sampling ratio r_f for base models are determined using a two-fold cross validation on the training set. First, we learn a baseline classifier with all the data and features. Then we search a list of pre-defined ratios for the minimal ones that can achieve at least 80% of the baseline performance on average. Finally, we have $r_d = 0.2, r_f = 0.1$ for the TREC collection and $r_d = 0.1, r_f = 0.1$ for the Consumer collection. We learn the base models using SVMs with a RBF kernel $K(x_i, x_j) = e^{-\rho\|x_i - x_j\|^2}$ because the image data are typically not linearly separable. ρ is set to 0.05 based on two-fold cross validation. Although SVMs are not considered as "weak classifiers" in general, learning SVMs on a small size of data samples and feature subspace can be quite unstable and thus unable to significantly reduce the training error. In this case, plugging SVMs into a boosting learning algorithm can be beneficial. In order to improve the robustness for estimating α^t, we apply a χ^2 test with a confidence interval of 2.5% to filter out the irrelevant base models for each label before its learning process. We run all the ensemble learning approaches up to 100 iterations. To increase robustness of performance retrieval reporting for each approach, we repeat the experiments for 10 times and report their average performance.

6.2.2 Experimental results

Table 2 summarizes the overall performance (in terms of Mean-AP), training time (in terms of seconds), number of base models and total model size on both multimedia collections for six learning algorithms, include a baseline algorithm which learns an independent model for each label from all the data examples and feature space (Baseline), a bagging algorithm which combines 10 bagging models learned via the baseline classifier for each label (Bagging), a random subspace algorithm that learns an independent model for each label with the same data/feature sampling ratios as the default setting (Subspace), a round-robin random subspace bagging algorithm (RSBag), a non-model-shared subspace boosting algorithm(NSBoost) and the model-shared subspace boosting algorithm(MSSBoost). The confidence interval on the classification performance are reported on a confidence level of 95%. The total model size is computed by multiplying the number of support vectors with the number of features. First, we can observe that RSBag can gain considerable improvement over the random subspace learning algorithm, which suggests that aggregating multiple subspace SVM classifiers can be helpful. NSBoost brings an further performance gain with the same number of base models and a slightly more computational intensive training process by selecting the most useful base models in a gradient way. Finally, MSSBoost outperforms all five other algorithms in both collections by means of allowing base models to be shared across labels. Its improvement is statistically significant over four out of all five methods, i.e., Baseline, Subspace, RSBag and NSBoost. It is worth pointing out that MSSBoost has a considerably shorter learning process and a much smaller model size than the baseline classifiers. For instance, it achieves a 60-fold speedup for the training process and a 40-fold reduction on the final model size on the TREC collection. This observation confirms the effectiveness and efficiency of MSSBoost.

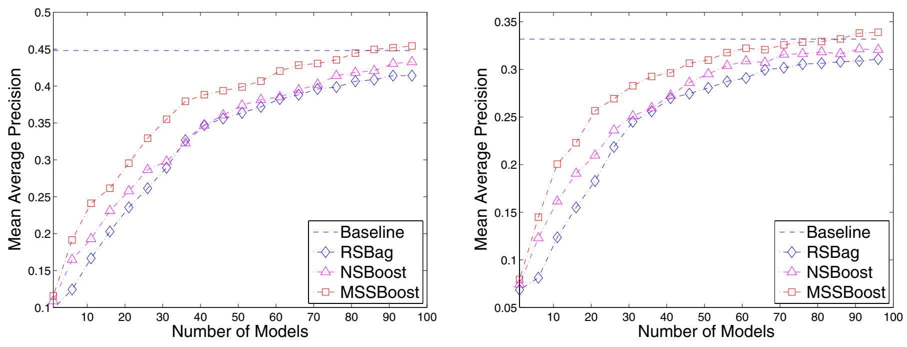

Figure 5: Learning curves of RSBag, NSBoost and MSSBoost as a function of the number of base models. The dashed line indicates the baseline performance. Left: TREC, Right: Consumer.

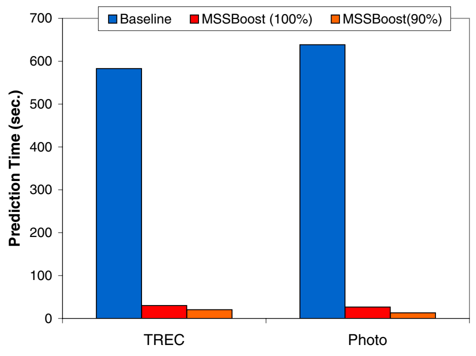

Figure 6: Comparison of the prediction time between Baseline and MSSBoost on all the testing examples. For MSSBoost, we identify the minimal number of base models needed to achieve 100% and 90% of baseline performance, and report the prediction time. Left: TREC, Right: Consumer.

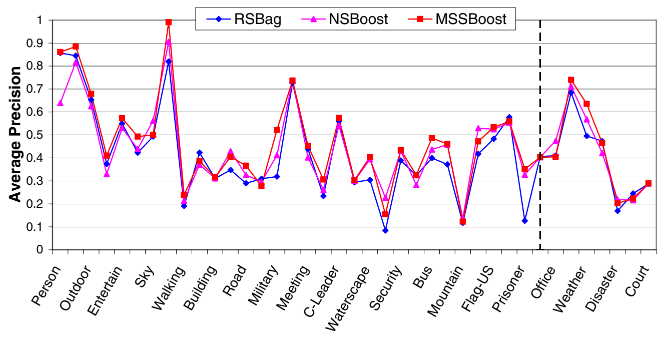

Figure 7: Average precision of RSBag, NSBoost and MSSBoost for each label in the TREC collection.

In Figure 5 we compare the learning curves of RSBag, NSBoost and MSSBoost for both multimedia collections when the number of base models is growing from 1 to 100. All three algorithms have better Mean-APs with more base models available. From the learning curves, we find that MSSBoost can bring even more noticeable improvement when the number of base models is small. For example, when using 35 base models in the TREC collection, the Mean-AP of MSBoost is about 7% higher (or relatively 20% higher) than that of NSBoost. From another viewpoint, MSSBoost can use a lower number of base models to reach the same level of classification performance as NSBoost and RSBag. For instance, MSSBoost only needs a minimum of 36 base models to produce a Mean-AP of 0.38 in TREC, while NSBoost and RSBag will require 54 models and 60 models to reach the same level. Figure 6 compares the prediction time of MSSBoost with the baseline algorithm. As we can observe, if MSSBoost is expected to have the same performance as baseline, it can reduce the prediction overhead by 20 times for the TREC collection and 30 times for the Consumer collection. If the expectation of MSSBoost is lowered to 90% of the baseline performance, the prediction overhead can be reduced by 35 times for the TREC collection and 60 times for the Consumer collection.

Figure 7 shows the label-by-label performance comparison between RSBag, NSBoost and MSSBoost in the TREC collection. We split the figure into two regions with a dashed vertical line. The right region contains all the labels that do not aggregate any base models from other labels, and the left region contains everything else. Inside each region, we sort the labels in a descending order of their frequencies. It can be found that the benefits of MSSBoost mainly comes from the labels that leverage base models from other related labels. Among them, the improvement of MSSBoost is quite consistent on the frequent labels, where it provides higher average precision than the other two methods on 8 out of the 9 most frequent labels. Overall, MSSBoost outperforms NSBoost on 30 (out of 39) labels and it outperforms RSBag on 34 (out of 39) labels, which are both statistically significant with a p-value < 0.01 using sign test.

Finally, Figure 8 illustrates the combination weight α_l^t and the base models h^t for the iteration t from 1 to 30. The selected base models in the earliest stage are mostly built from some general labels such as "cars", "studio", "leader" and so on. A close examination on the other dimension of the matrix indicates that most labels make use of the predictions from other related labels. For instance, the label "urban" can benefit from the models of its sub-concepts such as "car", "road", "airplane" and so on. On the other hand, the label "police" can benefit from its super-concept "military". These examples shows the advantages of model-sharing from two different perspectives.

7. CONCLUSION

In this paper, we proposed a boosting-type learning algorithm called model-shared subspace boosting (MSSBoost). It can automatically find, share and combine a number of

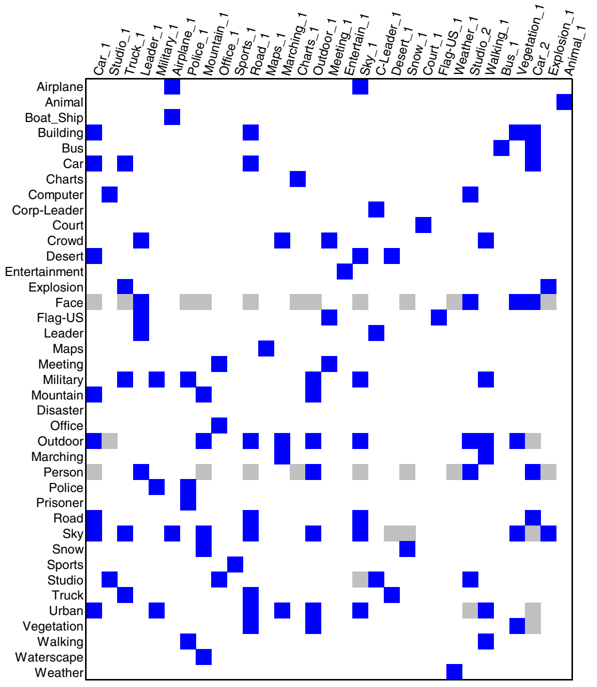

Figure 8: Matrix representation for the combination weight learned by MSSBoost. Each column indicates a selected base model with its associated label shown above. Each row indicates a label. Each grid corresponds to the weight of each base model, where blue means positive and grey means negative.

random subspace models across multiple labels. This algorithm is able to reduce the information redundancy in the label space by jointly optimizing the loss functions over all the labels. Meanwhile, they enjoy the advantage of being built on small base models, which are learned on a small number of bootstrap data samples and a randomly selected feature subspace. Our experimental results on a synthetic dataset and two real-world multimedia collections have demonstrated that the proposed MSSBoost algorithm can outperform the non-ensemble baseline classifiers with a significant speedup on the learning and prediction process. It can also use a smaller number of base models to achieve the same classification performance as its non-model-shared counterpart. As a by-product, the proposed algorithm can automatically discover the underlying context between multiple labels, which offers us additional insights to analyze the data collections. Our future work includes extending the proposed approach to other multi-label classification tasks such as text classification and protein function prediction.

Acknowledgement

This material is based upon work funded in part by the U. S. Government. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the U.S. Government. We want to thank NIST for providing the TRECVID annotation and our colleagues from IBM for annotating the consumer photo collection.

8. REFERENCES

[1] R. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Technical Report RC23462, IBM Research Center*, 45, 2004.

[2] K. Barnard, P. Duygulu, D. Forsyth, N. de Freitas, D. Blei, and M. Jordan. Matching words and pictures. *Journal of Machine Learning Research*, 3, 2002.

[3] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[4] L. Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 2001.

[5] R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[6] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes. Ensemble selection from libraries of models. In *Intl. Conf. of Machine Learning*, 2004.

[7] C. Chen, A. Liaw, , and L. Breiman. Using random forest to learn unbalanced data. Technical Report 666, Statistics Department, University of California at Berkeley, 2004.

[8] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997.

[9] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. Technical report, Dept. of Statistics, Stanford University, 1998.

[10] N. Ghamrawi and A. McCallum. Collective multi-label classification. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 195–200, New York, NY, USA, 2005. ACM Press.

[11] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning. Springer Series in Statistics.* Springer Verlag, Basel, 2001.

[12] T. K. Ho. The random subspace method for constructing decision forests. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(8):832–844, 1998.

[13] T. Joachims. Making large-scale support vector machine learning practical. In A. S. B. Schölkopf, C. Burges, editor, *Advances in Kernel Methods: Support Vector Machines.* MIT Press, Cambridge, MA, 1998.

[14] M. Naphade, J. R. Smith, J. Tesic, S.-F. Chang, W. Hsu, L. Kennedy, A. Hauptmann, and J. Curtis. Large-scale concept ontology for multimedia. *IEEE MultiMedia*, 13(3):86–91, 2006.

[15] A. Natsev, M. R. Naphade, and J. R. Smith. Semantic representation: search and mining of multimedia content. In *Proceedings of the 2004 ACM SIGKDD international conference*, pages 641–646, 2004.

[16] A. Natsev, M. R. Naphade, and J. Tešić. Learning the semantics of multimedia queries and concepts from a small number of examples. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 598–607, New York, NY, USA, 2005. ACM Press.

[17] S. Rosset. Robust boosting and its relation to bagging. In *Proceeding of the eleventh ACM SIGKDD international conference*, pages 249–255, New York, NY, USA, 2005.

[18] R. Schapire and Y. Singer. Boostexter: A system for multiclass multi-label text categorization. *Machine Learning*, 39(2), 2000.

[19] R. E. Schapire. Using output codes to boost multiclass learning problems. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 313–321, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[20] A. Smeaton and P. Over. TRECVID: Benchmarking the effectiveness of information retrieval tasks on digital video. In *Proc. of the Intl. Conf. on Image and Video Retrieval*, 2003.

[21] C. Snoek, M. Worring, J. Geusebroek, D. Koelma, and F. Seinstra. The mediamill TRECVID 2004 semantic viedo search engine. In *Proc. of TRECVID*, 2004.

[22] D. Tao, X. Tang, X. Li, and X. Wu. Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(7):1088–1099, 2006.

[23] A. Torralba, K. Murphy, and W. Freeman. Sharing visual features for multiclass and multiview object detection. In *IEEE Computer Vision and Pattern Recognition(CVPR)*, 2004.

[24] R. Yan and A. G. Hauptmann. Mining relationship between video concepts using probabilistic graphical model. In *Proceedings of IEEE International Conference On Multimedia and Expo (ICME)*, 2006.

[25] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proc. of the 14th ICML*, pages 412–420, 1997.

[26] J. Zhang, Z. Ghahramani, and Y. Yang. Learning multiple related tasks using latent independent component analysis. In *Neural Information Processing Systems (NIPS) 18*, 2005.