Hybrid Decision Tree Architecture Utilizing Local SVMs for Multi-Label Classification

Gjorgji Madjarov and Dejan Gjorgjevikj

Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University, Skopje, Macedonia
{gjorgji.madjarov,dejan.gjorgjevikj}@finki.ukim.mk

Abstract. Multi-label classification (MLC) problems abound in many areas, including text categorization, protein function classification, and semantic annotation of multimedia. Issues that severely limit the applicability of many current machine learning approaches to MLC are the large-scale problem and the high dimensionality of the label space, which have a strong impact on the computational complexity of learning. These problems are especially pronounced for approaches that transform MLC problems into a set of binary classification problems for which SVMs are used. On the other hand, the most efficient approaches to MLC, based on decision trees, have clearly lower predictive performance. We propose a hybrid decision tree architecture that utilizes local SVMs for efficient multi-label classification. We build decision trees for MLC, where the leaves do not give multi-label predictions directly, but rather contain SVM-based classifiers giving multi-label predictions. A binary relevance architecture is employed in each leaf, where a binary SVM classifier is built for each of the labels relevant to that particular leaf. We use several real-world datasets to evaluate the proposed method and its competition. Our hybrid approach on almost every classification problem outperforms the predictive performances of SVM-based approaches while its computational efficiency is significantly improved as a result of the integrated decision tree.

Keywords: multi-label classification, hybrid architecture.

1 Introduction

Single-label classification is concerned with learning from examples, each associated with a single label λ_i from a finite set of disjoint labels $L = \{\lambda_1, \lambda_2, ..., \lambda_Q\}$, $Q > 1$. The task is to learn a mapping l: $X \rightarrow L$ (where X denotes the example space) that assigns a single label to each example. For $Q > 2$, the task is referred to as a *multi-class classification*.

In multi-label classification (MLC), the task is to learn a mapping m: $X \rightarrow 2^L$. Each example $x \in X$ is associated to a set of labels $Y \subseteq L$. In contrast to multi-class classification, the labels are not assumed to be mutually exclusive, i.e., an example can be a member of more than one class. The labels in Y are called relevant and those in $L \backslash Y$ irrelevant for a given example.

E. Corchado et al. (Eds.): HAIS 2012, Part II, LNCS 7209, pp. 1–12, 2012.

Two major classes of algorithms for multi-label classification are decision-tree-based and SVM-based approaches. The first group is extremely efficient, but not very accurate while the second group, represented by the problem transformation SVM architectures [14] are very accurate, but can be computationally expensive, especially when labels abound. In this paper, we propose a novel hybrid architecture that integrates Decision Trees and Support Vector Machines for computationally efficient multi-label classification (ML-SVMDT) on large-scale problems with a large number of labels.

Section 2 surveys related previous work in multi-label learning. The architecture that we propose is presented in Section 3. Section 4 presents the experimental results that compare the performance of our architecture with the competing methods. The conclusions are given in Section 5.

2 Related Work

2.1 The Landscape of MLC Approaches

The issue of learning from multi-label data has recently attracted significant attention from many researchers. They are motivated by an increasing number of new applications, such as semantic annotation of images and video, functional genomics, music categorization into emotions, text classification, directed marketing and others. Many different approaches have been developed to solve the multi-label learning problems. Tsoumakas et al.[14] summarize them into two main categories: a) algorithm adaptation methods, and b) problem transformation methods. Algorithm adaptation methods extend specific learning algorithms to handle multi-label data directly. Examples include lazy learning [18], decision trees [2], neural networks [17], boosting [11], etc.

ML-C4.5 [2] is an adaptation of the well known C4.5 algorithm for multi-label learning. Clare et al. modified the formula of entropy calculation (equation 1) in order to solve multi-label problems. The modified entropy sums the entropies for each individual class label.

$$entropy(S) = -\sum_{i=1}^{N} (p(c_i) \log p(c_i) + q(c_i) \log q(c_i)) \quad (1)$$

where S is the set of examples, $p(c_i)$ is the relative frequency of class label c_i and $q(c_i) = 1 - p(c_i)$. They also allowed multiple labels in the leaves of the tree. Each leaf is represented by the most frequent set of class labels that are associated to the training examples that belong to that leaf. If more than 50% of the training examples in the leaf are labeled with a particular label then that label belongs to the set of most frequent labels.

ML-kNN [18] is based on the popular k Nearest Neighbors (kNN) lazy learning algorithm. The first step in this approach is the same as in kNN, i.e., retrieving the k nearest examples. It uses the maximum a posteriori principle in order to determine the label set of the test example, based on prior and posterior probabilities i.e. the frequency of each label within the k nearest neighbors.

Problem transformation methods, on the other hand, transform the multi-label learning problem into one or more single-label classification problems. The simplest and the most efficient strategy in terms of computational complexity in the multi-label setting is the one-against-all strategy, also referred to as the binary relevance (BR) method [14]. It addresses the multi-label learning problem by learning Q binary classifiers - one classifier for each label L. It transforms the original data set into Q data sets $S_{\lambda_i}, i = 1...Q$ that contain all examples of the original data set, labeled positively if the label set of the original example contained λ_i and negatively otherwise. For the classification of a new instance, each binary classifier predicts whether its label λ_i is relevant for the given example or not. Actually, BR outputs the union of the labels λ_i that are positively predicted by the Q classifiers. In the ranking scenario, the labels are ordered according to the probability associated to each label by the respective binary classifier.

A method closely related to the BR method is the Classifier Chain method (CC) proposed by Read et al. [10]. This method involves Q binary classifiers as in BR. Classifiers are linked along a chain where each classifier deals with the binary relevance problem associated with label $\lambda_i \in L$. The feature space of each link in the chain is extended with the 0/1 label associations of all previous links.

HOMER (Hierarchy Of Multi-label classifiERs) [15] is a computationally efficient multi-label classification method specifically designed for large multi-label datasets. HOMER constructs a hierarchy of multi-label classifiers, each one dealing with a much smaller set of labels compared to Q (the total number of labels) and a more balanced example distribution. This leads to improved predictive performance and also to linear training and logarithmic testing complexities with respect to Q. One of the main processes within HOMER is the even distribution of a set of labels into k disjoint subsets so that similar labels are placed together and dissimilar apart. The best predictive performance is reported using a balanced k means algorithm customized for HOMER [15].

Recently the most challenging issues in MLC are the high dimensionality of the label space and the problem of large datasets. These two problems can significantly influence on the computational complexity and the predictive performance of the MLC methods. Some proposed methods achieve higher computational efficiency at the cost of predictive accuracy, such as ML-kNN [18], ML-C4.5 [2], etc. These methods usually belong to the group of algorithm adaptation methods. Other proposed methods, based on problem transformation, use base classifiers with higher computational efficiency, such as Naive Bayes [7] [15], the one-layer perceptron [9], etc., in order to reduce the computational complexity.

2.2 Combining Decision Trees and SVMs

Several approaches that combine decision trees and SVMs have been proposed for binary and multi-class classification. For example, Kumar et al. [8] propose a method that combines decision trees and global SVM models (models learned on the whole dataset) for solving binary classification problems. Other approaches, such as [3], use the structure of the decision tree to organize/arrange SVM classifiers in its nodes in order to improve the computational efficiency and the

Table 1. The process of building the ML-SVMDT

procedure ML-SVMDT(S) **returns** tree
1: $(f^*, g^*, \mathcal{P}^*) = \text{BestFeature}(S)$
2: **if** $f^* \neq none$ **then**
3: **for each** $S_k \in \mathcal{P}^*$ **do**
4: $tree_k = \text{ML-SVMDT}(S_k)$
5: **return** node(f^*, $\bigcup_k \{tree_k\}$)
6: **else**
7: $localSVM = \text{trainBRModel}(S)$
8: **return** leaf($localSVM$)

procedure BestFeature(S)
1: $(f^*, g^*, \mathcal{P}^*) = (none, 0, \emptyset)$
2: **for each** feature f **do**
3: $\mathcal{P} =$ partition induced by f on S
4: $g = entropy(S) - \sum_{S_k \in \mathcal{P}} \frac{|S_k|}{|S|} entropy(S_k)$
5: **if** $(g > g^*) \wedge \text{Acceptable}(f, \mathcal{P})$ **then**
6: $(f^*, g^*, \mathcal{P}^*) = (f, g, \mathcal{P})$
7: **return** $(f^*, g^*, \mathcal{P}^*)$

predictive performance. Boosting ensemble of support vector machines for multi-class classification was proposed in [13]. Gama [5] proposed functional trees for multi-class classification and regression problems. However, none of these approaches deal with MLC problems.

3 Integration of Decision Trees and SVMs

In this paper, we propose a novel hybrid approach for computationally efficient multi-label classification that combines the algorithm adaptation method ML-C4.5 [2] and the problem transformation method Binary Relevance (BR) [14]: The latter uses SVMs as base classifiers for solving the partial binary classification problems. The main idea of our approach is to use the advantages of both methods - the low computational complexity of ML-C4.5 and the predictive accuracy of the BR architecture with SVM classifiers.

One approach to achieve effective and computationally efficient multi-label classification is to partition the global classification problem first and then learn local classifiers for each of those partitions (subproblems) separately. In the prediction phase, first one tries to determine the partition to which a test example belongs, and then to classify the example using a local classifier trained using the examples belonging to that partition only. The logic behind this approach is that the "neighbors" of a test example (training examples that belong to the same partition as the test example), would be able to provide more accurate information about it faster.

We propose a novel hybrid architecture that introduces local models for solving multi-label learning problems, based on SVM classifiers. Our approach combines the ML-C4.5 method for partitioning the input feature space and the BR method utilizing SVMs as base classifiers for local classification. The main idea is to use the advantages of both methods, in order to build an architecture that will improve the predictive performance and the computational efficiency.

Throughout the literature, BR is often mentioned, but consistently criticized on account of its assumption of label independence. Namely, during the transformation process, BR ignores label correlations that exist in the training data. In order to reduce these inconsistencies in the multi-label prediction of the BR

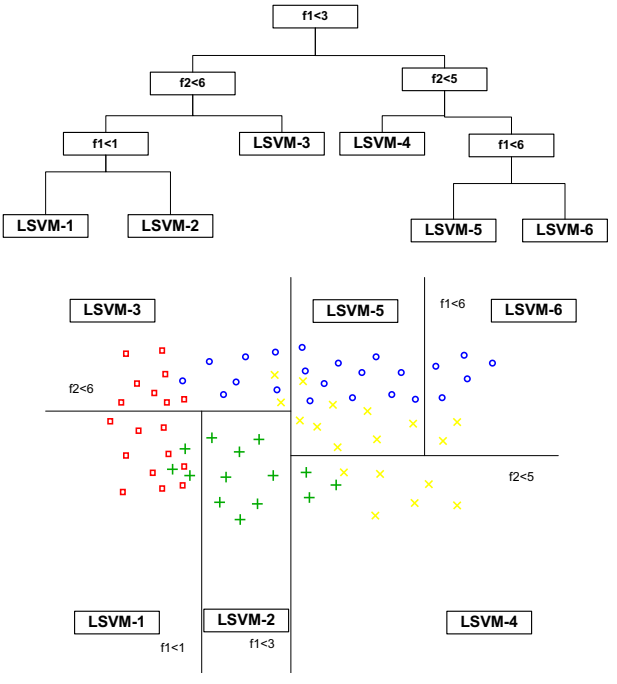


Fig. 1. ML-SVMDT splits the original dataset into subsets and builds a local SVM model (LSVM) for each partition

method, we employed the ML-C4.5 method. In this context, the ML-C4.5 method actually tries to separate the training examples in groups where the label inconsistency will be eliminated. It finds the correlation between labels and according to that correlation it splits the global problem into several local subproblems.

The procedure (ML-SVMDT(S)) of building the architecture is outlined in Table 1. It takes as input a set of examples (S) and outputs a tree. The process of building starts at the root of the tree with choosing one feature (f) of the data that most effectively splits the dataset of examples into subsets ($\mathcal{P}$). The criterion is the normalized information gain (difference in entropy g) that results from choosing a feature for splitting the data (line 6 of BestFeature procedure in Table 1). The feature with the highest normalized information gain is chosen to make the decision. The process continues recursively in each node until all the examples are labeled with the same labels or none of the features provide any information gain. Tree building also stops if some predetermined minimal number of examples per node is reached. If no acceptable feature can be found or some predetermined minimal number of examples per node is reached (Acceptable($f, \mathcal{P}$)), the process of splitting the dataset stops and the corresponding node is declared as a leaf. After that, we try to solve the "new" problem defined on the examples in the leaf by using a local model. This means that every leaf of the decision

tree is replaced with one local model built by the BR method on the training examples that belong to the corresponding leaf, using SVMs as base classifiers (Figure 1).

The testing for each test example starts at the root of the tree. The decision tree transfers the example to exactly one leaf of the tree according to its features. The final decision about the labeling of the test example is performed by the local model in the corresponding node consisting of SVMs. Each test example consults only one local model in order to be classified. The testing time for each test example is the sum of the time needed to sort the example through the decision tree and the time needed for the corresponding local model to make a decision.

4 Experiments

In this section the performance of the proposed method is compared to the performances of the competing method for multi-label learning in domains with large number of labels (HOMER [15]) and two additional state of the art methods (Classifier Chains - CC [10] and ML-kNN [18]). We also compare it to the two methods that it integrates, the ML-C4.5 [2] method and the BR [14] method. The performance is measured in terms of five different multi-label evaluation measures (two example based measures - Hamming loss and precision, two label based measures - micro precision and macro precision and one ranking based measure - average precision) [16]. In the results we also report the training and the testing times of all methods (measured in seconds).

4.1 Datasets and Experimental Setup

Five different multi-label classification problems were addressed in our experiments. The predictive performance in terms of the measures mentioned above and the training and testing times were recorded for every method for each classification problem. The complete description of the datasets in terms of application domain (*domain*), the number of training (*#tr.e.*) and test (*#t.e.*) examples, the number of features (D), the total number of labels (Q) and label cardinality (l_c) [14] are shown in Table 2.

We strived to include a considerable variety and scale of multi-label datasets. In total we use five datasets, with dimensions ranging from 101 to 983 labels,

Table 2. Dataset description

	domain	*#tr.e.*	*#t.e.*	D	Q	l_c
corel5k [4]	image	4500	500	499	374	3.52
mediamill [12]	video	30993	12914	120	101	4.38
bibtex [7]	text	4880	2515	1836	159	2.40
delicious [15]	text	12920	3185	500	983	19.02
bookmarks [7]	text	60000	27856	2150	208	2.03

and from less than 5,000 examples to almost 90,000. The datasets are roughly ordered by complexity ($\#tr.e. \times D \times Q$).

The training and testing of the proposed method were performed using a custom developed application that uses the MULAN library[1] for the machine learning framework Weka [6]. We implemented the ML-C4.5 method within the same library, while HOMER and BR was already implemented in MULAN. For the CC method we used the MEKA[2] extension for the WEKA framework. All experiments were performed on a server with an Intel Xeon processor at 2.50GHz on 64GB of RAM with the Fedora 14 operating system.

The LIBSVM library [1], and in particular SVMs with a radial basis kernel, were used for solving the binary classification problems for all datasets in the BR, CC, HOMER and the proposed method. The kernel parameter *gamma* and the penalty C for the datasets for each method were determined by 10-fold cross validation using only the training sets. The values $2^{-15}, 2^{-13}, ..., 2^{1}, 2^{3}$ were consider for *gamma* and $2^{-5}, 2^{-3}, ..., 2^{13}, 2^{15}$ for the penalty C. After determining the best parameters values for each method on every dataset the classifiers were trained using all available training examples and were evaluated by recognizing all test examples from the corresponding dataset.

The ML-C4.5 method uses subtree raising with a pruning confidence of 0.25 as a post pruning strategy in all classification problems. The number of neighbors in the ML-kNN method for each dataset was determined from the values 6 to 20 with step 2 for which the best results were obtained. To define the subsets of labels in each level of the hierarchy in HOMER, we used the balanced k means algorithm proposed by the original authors. This algorithm requires one parameter to be configured: number of subsets k. Five different values (2-6) were considered in the experiments for this parameter. These values were used by the authors [15]. For all methods, the best obtained results are presented in the Results subsection. Additionally, to access the dependence of the predictive performance and the computational complexity of ML-SVMDT on the minimal number of examples in the leaves of the decision tree, we tried six different values of the minimal number of examples (30-80) in the leaves. Overall, the proposed architecture obtained the best predictive performance when the minimal number of examples in the leaves was set to 70 and these results are presented in the Results subsection.

4.2 Results

Table 3 gives the predictive performance in terms of the five evaluation measures, the training and testing times for each method on each of the datasets. The first column of the table lists the evaluation measures, while the second column lists the classification problems. The remaining columns show the performance of each method for every dataset. The best results per dataset in the table are shown in boldface. DNF in the results indicates that the experiment Did Not Finish

[1] `http://mulan.sourceforge.net/`
[2] `http://meka.sourceforge.net/`

Table 3. The predictive performance in terms of the five evaluation measures, the training and the testing times measured in seconds

	datasets	BR	CC	ML-C4.5	ML-kNN	HOMER	ML-SVMDT
Hamming loss	**corel5k**	0.017	0.017	0.010	**0.009**	0.012	**0.009**
	mediamill	0.032	0.032	0.044	**0.031**	0.038	0.032
	bibtex	**0.012**	**0.012**	0.016	0.014	0.014	**0.012**
	delicious	**0.018**	**0.018**	0.019	**0.018**	0.022	**0.018**
	bookmarks	DNF	DNF	**0.009**	**0.009**	DNF	**0.009**
Precision	**corel5k**	0.042	0.042	0.005	0.035	**0.317**	0.127
	mediamill	0.731	**0.741**	0.056	0.724	0.597	0.727
	bibtex	**0.515**	0.508	0.123	0.254	0.472	0.484
	delicious	0.443	0.399	0.001	0.424	0.369	**0.486**
	bookmarks	DNF	DNF	0.271	0.218	DNF	**0.281**
mi. Precision	**corel5k**	0.061	0.061	0.160	**0.730**	0.308	0.664
	mediamill	0.742	**0.753**	0.597	0.739	0.569	0.749
	bibtex	0.753	0.744	0.359	**0.819**	0.547	0.789
	delicious	0.658	0.660	0.000	0.651	0.396	**0.662**
	bookmarks	DNF	DNF	0.632	0.850	DNF	**0.855**
ma. Precision	**corel5k**	0.052	0.053	0.004	0.031	0.044	**0.055**
	mediamill	0.112	0.144	0.046	**0.308**	0.107	0.258
	bibtex	0.528	**0.539**	0.128	0.192	0.391	0.495
	delicious	0.299	0.303	0.000	0.134	0.154	**0.312**
	bookmarks	DNF	DNF	0.292	0.414	DNF	**0.485**
Avg. Precision	**corel5k**	0.303	0.293	0.196	0.266	0.222	**0.306**
	mediamill	0.686	0.672	0.669	**0.703**	0.583	0.698
	bibtex	0.597	**0.599**	0.392	0.349	0.407	0.563
	delicious	0.351	0.343	0.321	0.326	0.231	**0.362**
	bookmarks	DNF	DNF	0.378	0.381	DNF	**0.421**
Training times	**corel5k**	926	1225	369	389	771	**274**
	mediamill	85468	100435	2030	**1094**	78195	9015
	bibtex	11013	12434	566	**124**	2869	767
	delicious	57053	84903	2738	**236**	21218	1168
	bookmarks	DNF	DNF	**4039**	15990	DNF	53737
Testing times	**corel5k**	25	31	**1**	45	14	9
	mediamill	6152	6125	**1**	477	6079	398
	bibtex	654	661	**7**	64	155	84
	delicious	2045	1872	**19**	55	816	189
	bookmarks	DNF	DNF	**21**	4084	DNF	4189

within one week under the available resources. Training time of the ML-kNN method is the time needed for calculating the posterior probability of each label within the k nearest neighbors.

Overall, the results show that ML-SVMDT outperforms HOMER and ML-C4.5 on all five datasets in terms of the five evaluation measures. The difference in the predictive performances between ML-SVMDT and HOMER is more evident for the larger datasets (bibtex and delicious). Compared to the BR, CC and the ML-kNN methods, ML-SVMDT shows better performance in terms of the ranking based measure for all datasets, except for the bibtex dataset where BR and CC show slightly better results. For the example and label based measures the proposed method outperforms BR, CC and ML-kNN for the delicious dataset and shows comparable performance for the corel5k, mediamill and bibtex datasets.

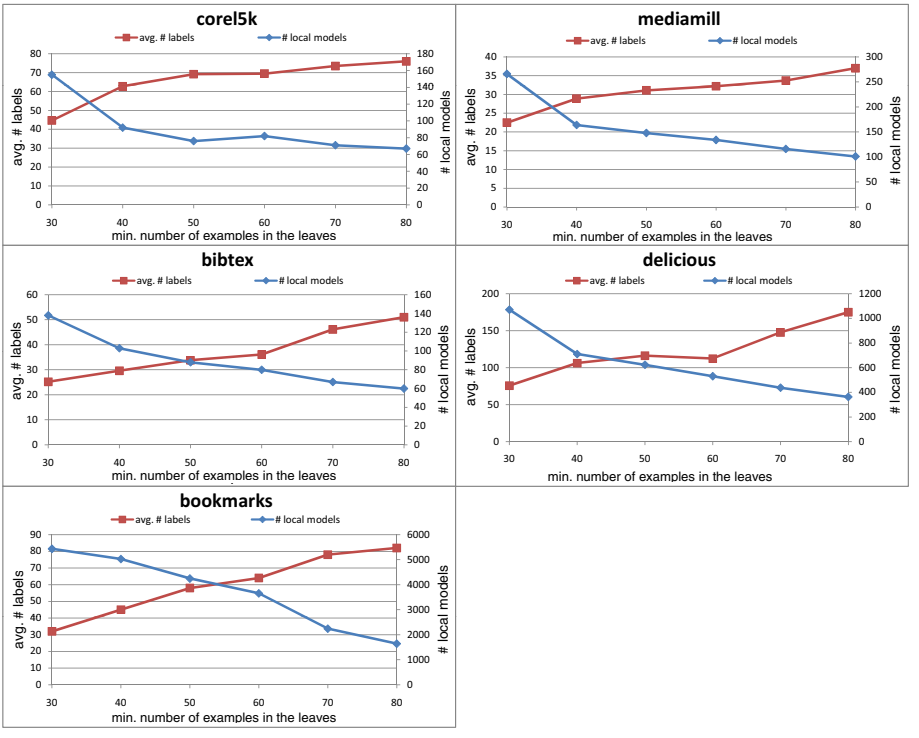
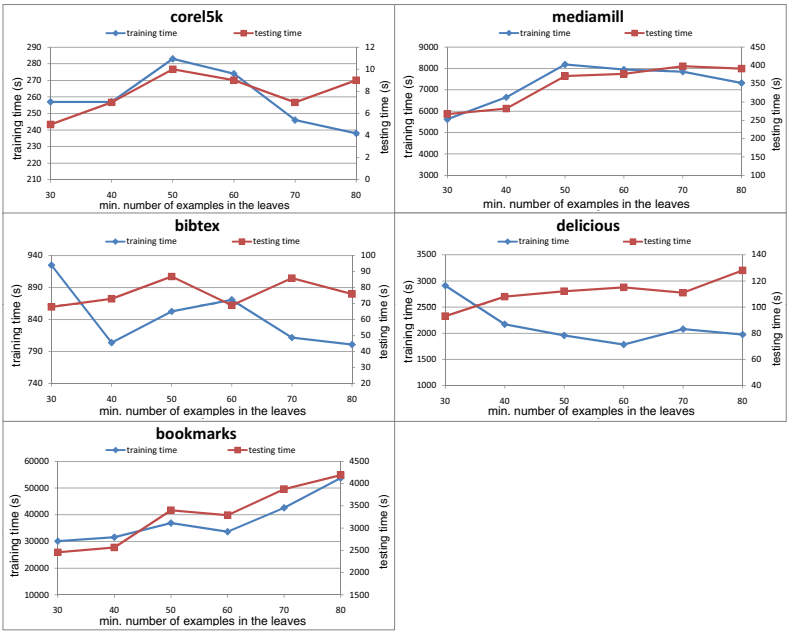
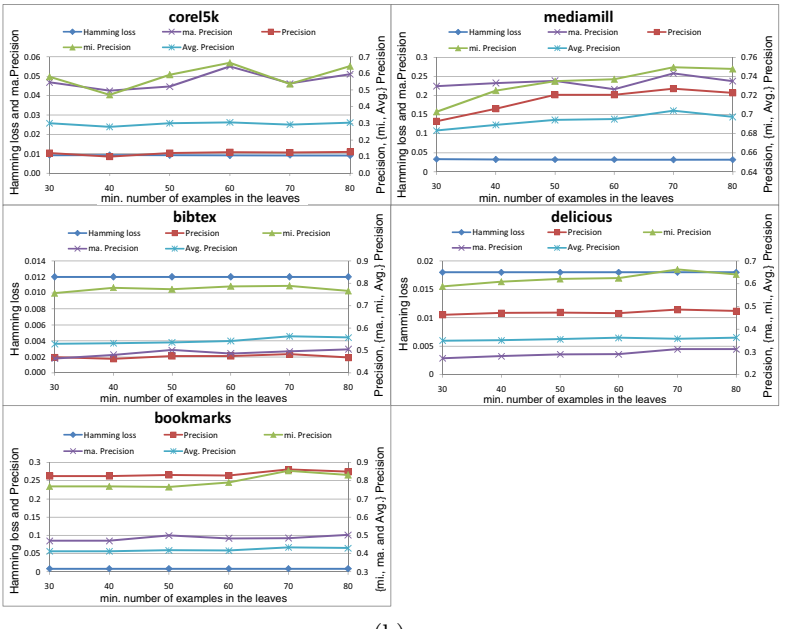


Fig. 2. The average number of labels per local model and the number of local models of ML-SVMDT as functions of the number of the minimal number of examples in the leaves

The results in terms of training and testing speed show that the proposed method is 2 to 20 times faster in the training phase and 1.4 to 10 times faster in the testing phase than the HOMER method. The computational efficiency of ML-SVMDT is even higher compared to the BR and CC methods (5 to 35 times in the training phase and 3 to 25 times in the testing phase). ML-kNN is the fastest in the training phase for the mediamill, bibtex and delicious datasets, while ML-C4.5 shows the best training time for the bookmarks dataset. The proposed architecture also shows higher computational efficiency than the ML-C4.5 method in the training phase for the datasets with a larger number of labels (corel5k and delicious) as a result of the post-pruning method used in the ML-C4.5 algorithm that gives additional computational complexity. ML-SVMDT uses only the minimal number of examples in the leaves of the tree as a pre-pruning method that controls the size of the tree: After a node reaches the minimal number of examples, no further branching of the decision tree is allowed. On the other hand, ML-C4.5 is faster than ML-SVMDT in the training phase for the other three datasets (mediamill, bibtex and bookmarks) and it is the fastest in the testing phase for all datasets.



(a)



(b)

Fig. 3. (a) The training and testing times and (b) the predictive performance of ML-SVMDT as functions of the number of the minimal number of examples in the leaves

In the case of the bookmarks dataset, only the ML-C4.5, ML-kNN and the ML-SVMDT methods perform satisfactorily under the experiment setup. The other methods suffer problem of higher computational complexity. Bookmarks dataset is much larger than those typically approached in the literature. On this dataset, ML-SVMDT shows the best predictive performance and the overall advantage in time costs compared to the other methods.

The dependence of the predictive performance and the training and testing times of the proposed architecture on different values of the minimal number of examples in the leaves graphically are shown on Figures 3(a) and 3(b), for each dataset separately. It should be noticed that the predictive performance of the proposed architecture strongly depends on this parameter. Overall, the best predictive performance were obtained when the minimal number of examples in leaves was set to 70. For the bookmarks dataset, some predictive performance further improved when the minimal number of examples in leaves rose to 80 as a result of the larger number of examples in the dataset compared to the other datasets. On Figure 2, the dependence of the average number of labels per local model (avg. # labels) and the number of local models (# local models) generated in the ML-SVMDT architecture on the minimal number of examples in the leaves are shown graphically for each dataset. These two parameters are closely related to the number of examples in the leaves: By increasing the minimal number of examples in the leaves, the average number of labels per local model increases, while the number of local models generated by ML-SVMDT decreases.

5 Conclusions

We propose a novel hybrid architecture that integrates Decision Trees and SVMs for computationally efficient multi-label classification. The architecture combines the algorithm adaptation method ML-C4.5 and the problem transformation method Binary Relevance: The latter uses SVMs as base classifiers for solving the binary classification problems.

The proposed architecture is compared to the BR method, CC, ML-C4.5, ML-kNN and the HOMER method on five different real-world datasets. Among the six competing methods, ML-C4.5 is the fastest one in the prediction phase. ML-kNN shows lower computational complexity in the training and slightly higher computational complexity in the prediction phase compared to the ML-C4.5 method, but, it is better in terms of the predictive performance. ML-SVMDT shows slightly higher training times and comparable, but slightly smaller testing times than ML-kNN, while showing better predictive performance in terms of the example and ranking based evaluation measures. In terms of the label based measures ML-SVMDT outperforms the ML-kNN method for the two largest datasets (delicious and bookmarks) and corel5k (the second dataset ordered by label dimensionality) and shows comparable results for the other two datasets. Compared to the BR and CC methods, it shows slightly better predictive performance but significantly higher computational efficiency. ML-SVMDT also clearly outperforms the HOMER method in both predictive performance and speed.

Despite other methods, ML-SVMDT can achieve better predictive performance and is efficient enough to scale up to very large problems.

References

1. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2001), software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
2. Clare, A., King, R.D.: Knowledge Discovery in Multi-label Phenotype Data. In: Siebes, A., De Raedt, L. (eds.) PKDD 2001. LNCS (LNAI), vol. 2168, pp. 42–53. Springer, Heidelberg (2001)
3. Dong, G.M., Chen, J.: Study on support vector machine based decision tree and application. In: Proc. of the 5th International Conference on Fuzzy Systems and Knowledge Discovery, pp. 318–322 (2008)
4. Duygulu, P., Barnard, K., de Freitas, J.F.G., Forsyth, D.: Object Recognition as Machine Translation: Learning a Lexicon for a Fixed Image Vocabulary. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2353, pp. 97–112. Springer, Heidelberg (2002)
5. Gama, J.: Functional trees. Machine Learning 55, 219–250 (2004)
6. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. SIGKDD Explorations 11, 10–18 (2009)
7. Katakis, I., Tsoumakas, G., Vlahavas, I.: Multilabel Text Classification for Automated Tag Suggestion. In: Proc. of the ECML/PKDD Discovery Challenge (2008)
8. Kumar, A.M., Gopal, M.: A hybrid svm based decision tree. Pattern Recognition 43, 3977–3987 (2010)
9. Mencía, E.L., Park, S.H., Fürnkranz, J.: Efficient voting prediction for pairwise multilabel classification. Neurocomputing 73, 1164–1176 (2010)
10. Read, J., Pfahringer, B., Holmes, G.: Multi-label Classification Using Ensembles of Pruned Sets. In: Proc. of the 8th IEEE International Conference on Data Mining, pp. 995–1000 (2008)
11. Schapire, R.E., Singer, Y.: Boostexter: A boosting-based system for text categorization. Machine Learning 39, 135–168 (2000)
12. Snoek, C.G.M., Worring, M., van Gemert, J.C., Geusebroek, J.M., Smeulders, A.W.M.: The challenge problem for automated detection of 101 semantic concepts in multimedia. In: Proc. of the 14th Annual ACM International Conference on Multimedia, pp. 421–430 (2006)
13. Ting, K.M., Zhu, L.: Boosting Support Vector Machines Successfully. In: Benediktsson, J.A., Kittler, J., Roli, F. (eds.) MCS 2009. LNCS, vol. 5519, pp. 509–518. Springer, Heidelberg (2009)
14. Tsoumakas, G., Katakis, I.: Multi Label Classification: An Overview. International Journal of Data Warehouse and Mining 3(3), 1–13 (2007)
15. Tsoumakas, G., Katakis, I., Vlahavas, I.: Effective and Efficient Multilabel Classification in Domains with Large Number of Labels. In: Proc. of the ECML/PKDD Workshop on Mining Multidimensional Data, pp. 30–44 (2008)
16. Tsoumakas, G., Katakis, I., Vlahavas, I.: Mining multi-label data. In: Data Mining and Knowledge Discovery Handbook, pp. 667–685. Springer, Heidelberg (2010)
17. Zhang, M.L., Zhou, Z.H.: Multi-label neural networks with applications to functional genomics and text categorization. IEEE Transactions on Knowledge and Data Engineering 18(10), 1338–1351 (2006)
18. Zhang, M.L., Zhou, Z.H.: Ml-knn: A lazy learning approach to multi-label learning. Pattern Recognition 40(7), 2038–2048 (2007)