ECML PKDD 2009
European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases

LEARNING FROM MULTI-LABEL DATA AT ECML PKDD 2009

MLD'09

September 7, 2009

Bled, Slovenia

Preface

This volume contains research papers accepted for presentation at the 1st International Workshop on Learning from Multi-Label Data (MLD'09), which will be held in Bled, Slovenia, at September 7, 2009 in conjunction with ECML/PKDD 2009 .

MLD'09 is devoted to multi-label learning, which is an emerging and promising research topic of machine learning. In multi-label learning, each example is associated with multiple labels simultaneously, which therefore encompasses traditional supervised learning (single-label) as its special case. Multi-label learning is related to various machine learning paradigms, such as classification, ranking, semi-supervised learning, active learning, multi-instance learning, dimensionality reduction, etc.

Initial attempts on multi-label learning date back to 1999 with works on multi-label text categorization. In recent years, the task of learning from multi-label data has been addressed by a number of methods adapted from various popular learning techniques, such as neural networks, decision trees, k-nearest neighbors, kernel methods, ensemble methods, etc. More impressively, multi-label learning has manifested its effectiveness in a diversity of real-world applications, such as image/video annotation, bioinformatics, web search and mining, music categorization, collaborative tagging, directed marketing, etc.

The goal of MLD'09 is to provide an interactive forum for researchers and practitioners interested in multi-label learning to share their minds. A total of 16 submissions were received and each paper was rigorously reviewed by two PC members. Based on the returned reviews, 12 papers were accepted for presentation. The papers discuss state-of-the-art as well as new research directions of multi-label learning.

We greatly appreciate the many people who contributed to the successful organization of MLD'09. First of all, we wish to sincerely thank all the authors who submitted their work for consideration. We would also like to thank the Program Committee members as well as external reviewers for their excellent efforts in keeping the review process at high quality. Last, but not least, we would like to thank everybody who helped the organization of MLD'09 and the production of this volume, especially ECML/PKDD 2009 General Chair, Dunja Mladenic, and Workshop Chair, Rayid Ghani.

August 2009

Grigorios Tsoumakas
Min-Ling Zhang
Zhi-Hua Zhou

Organization

Workshop Co-Chairs

Grigorios Tsoumakas	Aristotle University of Thessaloniki
Min-Ling Zhang	Hohai University
Zhi-Hua Zhou	Nanjing University

Program Committee

Hendrik Blockeel	Katholieke Universiteit Leuven
Johannes Fürnkranz	TU Darmstadt
Shantanu Godbole	IBM Research
Xian-Sheng Hua	Microsoft Research Asia
Eyke Hüllermeier	Philipps-Universität Marburg
Ioannis Katakis	Aristotle University of Thessaloniki
Dragi Kocev	Jožef Stefan Institute
Jose M. Peña	Universidad Politécnica de Madrid
Bernhard Pfahringer	University of Waikato
Fadi Thabtah	University of Huddersfield
Jieping Ye	Arizona State University
Kai Yu	NEC Laboratories America, Inc.
Shipeng Yu	Siemens Medical Solutions USA, Inc.

Additional Referees

Beau Piccart	Katholieke Universiteit Leuven
Jesse Read	University of Waikato
Jan Struyf	Katholieke Universiteit Leuven
Jingdong Wang	Microsoft Research Asia

Table of Contents

Evaluation of Distance Measures for Hierarchical Multi-Label Classification in Functional Genomics

Darko Aleksovski, Dragi Kocev, and Sašo Džeroski

Department of Knowledge Technologies, Jozef Stefan Institute
Jamova cesta 39, 1000 Ljubljana, Slovenia
{Darko.Aleksovski, Dragi.Kocev, Saso.Dzeroski}@ijs.si

Abstract. Hierarchical multi-label classification (HMLC) is a variant of classification where instances may belong to multiple classes that are organized in a hierarchy. The approach we used is based on decision trees and is set in the predictive clustering trees framework (PCTs), which is implemented in the CLUS system. In this work, we are investigating how different distance measures for hierarchies influence the predictive performance of the PCTs. The distance measures that we consider include weghted Euclidean distance, Jaccard, SimGIC and ImageCLEF distance. We use datasets from the area of functional genomics to evaluate the performance of the PCTs with different distances. The datasets describe different functions of the genes in the genomes of two well-studied organisms: S. Cerevisiae and A. Thaliana. We use precision-recall curves as an evaluation metric for the predictive performance. The results from the Friedman test for statistical significance suggest that there is no statistical significance in the performance.

1 Introduction

Hierarchical multi-label classification (HMLC) is an extension of binary classification where an instance can be labeled with multiple classes that are organized in a hierarchy. Additionally, when an instance is assigned to some class it should also be assigned to all its superclasses. The main applications of HMLC are in the areas of gene function prediction [1, 2], text classification [3] and image classification [4].

There are two general approaches for solving the HMLC task: decomposing this task to simpler single-target tasks and solving them with basic classification approaches or using the hierarchical structure and trying to make predictions for the whole hierarchy. An example for the first approach is learning a binary classifier for each class and an example for the second approach is to learn a single model which predicts all the classes simultaneously. The second group of algorithms has some advantages over the first group [5–7]. First, they exploit the dependencies between the components and as a result have better predictive performance. Second, they are more efficient: it can easily happen that the number of components in the output is very large (e.g., hierarchies in functional genomics) in which case running a learning algorithm for each component is not feasible. Third, they produce a single model valid for the structure as a whole, as compared to the many models, each valid just for one given component: the single model is usually much more concise.

In this study, we focus on the latter approach: we learn a single Predictive Clustering Tree [6] to make a prediction for the complete hierarchy. The PCTs were extended to the HMLC task by Vens at al. [2], and they use weighted Euclidean distance as a distance measure between two hierarchies. Here, we consider three additional distance measures (Jaccard distance [13], SimGIC [8] and ImageCLEF [9]). We implemented the distance measures in the CLUS system and we evaluated them on several datasets from functional genomics.

The remainder of this paper is organized as follows: Section 2 describes the PCTs algorithm and the proposed distance measures and Section 3 presents the datasets that we used for evaluation. Section 4 gives the experimental design, while Section 5 presents the obtained results. Finally, conclusions and points for further work are presented in Section 6.

2 Methodology

2.1 Predictive Clustering Trees

The approach we use is based on decision trees and is set in the predictive clustering trees (PCTs) framework. This framework views a decision tree as a hierarchy of clusters, where the top node correspond to a cluster containing all data, which are recursively partitioned into smaller clusters while building the tree from top to bottom. The PCT framework is implemented in the CLUS system (available at `http://www.cs.kuleuven.be/~dtai/clus`).

PCTs can be constructed with a standard "top-down induction of decision trees" (TDIDT) algorithm. The heuristic that is used for selecting the tests is the reduction in variance caused by partitioning the instances. Maximizing the variance reduction maximizes cluster homogeneity and improves predictive performance. With appropriate instantiation of the variance and prototype function the PCTs can handle different types of data, e.g., multiple targets [11] or time series [12]. A detailed description of the PCT framework can be found in [6].

In the remainder of this sub-section, we explain how PCTs were instantiated for the HMLC task, namely we present the internal representation of the hierarchy, annotation of the examples, making a prediction and we give an example of PCT for HMLC. The hierarchy is represented as a 0/1 vector: if a given example is labeled with some label, then for that label the value in the vector is set to 1, otherwise it is set to 0. The annotation scheme is presented in Figure 1. The example is annotated with the following labels: B, B.1, C, D, D.2 and D.3. If an example belongs to a node, then it belongs also to all the node's parents.

The reduction of variance is calculated using the following equation:

$$Var(S) = \frac{\sum_i d(v_i, \bar{v})^2}{|S|} \qquad (1)$$

where S denotes the set of examples over which the variance is calculated, $\bar{v}$ is the mean label, and v_i is a label of the example. The sum goes over all possible labels. The mean label is calculated as the mean of the vectors of the examples from S, in that node.

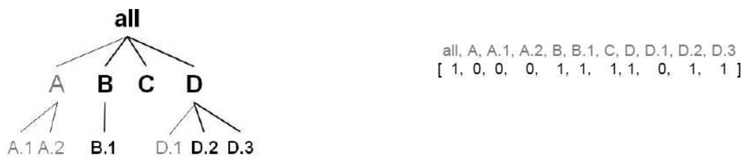


Fig. 1. A hierarchy (left) with an example annotated to it (subset of the hierarchy shown bold); the example's vector representation (right).

Different distance measures can be used in equation 1. In the original implementation from [2], the Euclidean distance is used for PCT induction. In this work, we use three other distances that can be used in the context of HMLC. We present and explain the distances in the next sub-sections.

The PCTs at every leaf of the tree contain probabilities (a probability vector) of an instance belonging to each class in the hierarchy. To obtain a prediction, a threshold is applied to the probability vector. If a given label has a bigger probability than the threshold, the example is annotated with that label and its parents. An example of a PCT for HMLC is presented in Figure 2. It looks like an ordinary decision tree, but in the leaves, instead of the majority class, it contains as prediction the annotation for the examples from that node. Note that for some of the leaves have prediction: "none". This is because no annotations could be assigned for the used threshold value (i.e., the probabilities for example belonging to the classes are lower than the specified threshold).

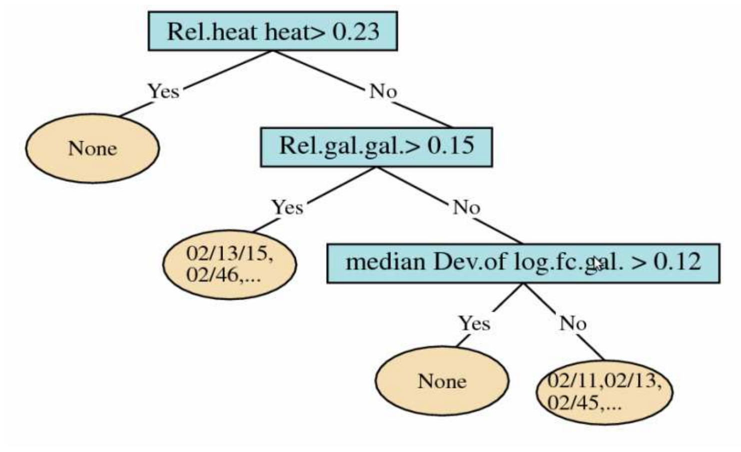


Fig. 2. An example PCT for HMLC, obtained with a given threshold, for the 'church' dataset with FunCat annotation.

2.2 Weighted Euclidean distance

The Euclidean distance is a well known distance measure. In order to include knowledge about the hierarchy, Vens et al. [2] have introduced a weighting scheme that depends on the depth of the node in the hierarchy. The weighted Euclidean distance can be calculated using the following equation:

$$d(v_1, v_2) = \sqrt{\sum_i w(c_i)(v_{1,i} - v_{2,i})^2} \tag{2}$$

where $v_{k,i}$ is the i'th component of the class vector v_k of an instance x_k. The function $w(c)$ is denoted as the weighting scheme and the default instantiation here is to apply a weight to each class label c according to the depth of this class in the hierarchy (e.g., $w(c) = w_0^{depth(c)}$ with $0 < w_0 < 1$). With this parameter, the user can control the influence of the top classes on the distance.

Let us consider two examples: (x_1, S_1) and (x_2, S_2), which are annotated with the hierarchy from Figure 1: S_1 = {B, B.1, C, D, D.2, D.3} and S_2 = {D, D.2, D.3}. Using the vector representation presented above, the weighted Euclidean distance is:

$$d(S_1, S_2) = \sqrt{w_0 + w_0^2 + w_0} \tag{3}$$

The weighting function described here is only one of the possible weighting schemes that can be used. Others weighting schemes are described in [2] and it is recommended to use weighting.

2.3 Jaccard distance

The Jaccard distance [13] (which can also be found in the literature as Union-intersection distance/score) can be calculated using the following equation:

$$d_{Jaccard}(v_1, v_2) = 1 - \frac{\sum_{c \in labels(v_1) \cap labels(v_2)} w(c)}{\sum_{c \in labels(v_1) \cup labels(v_2)} w(c)} \tag{4}$$

where v_1 and v_2 are class vectors, $labels(v)$ presents the elements from v, c is a class node from v_k. This distance actually is taking into account the ratio between the sum of the weights of the joint annotations and the sum of the weights of the annotations of both examples. As in the case of weighted Euclidean distance, we use the same exponential weighting scheme.

Let us consider the same example as for the weighted Euclidean distance. The Jaccard distance for the two examples (x_1, S_1) and (x_2, S_2) will be:

$$d(S_1, S_2) = 1 - \frac{w_0^0 + w_0 + w_0^2 + w_0^2}{w_0^0 + w_0 + w_0^2 + w_0 + w_0 + w_0^2 + w_0^2} = 1 - \frac{1 + w_0 + 2w_0^2}{1 + w3_0 + 3w_0^2} \tag{5}$$

2.4 SimGIC distance

The Similarity for Graph Information Content (SimGIC) distance [8] is similar to the Jaccard distance, but instead of summing the weights of the labels, it sums up their information content [2].

$$d_{SimGIC}(v_1, v_2) = 1 - \frac{\sum_{c \in labels(v_1) \cap labels(v_2)} IC(c)}{\sum_{c \in labels(v_1) \cup labels(v_2)} IC(c)} \tag{6}$$

The variables here are the same as for the Jaccard distance, and $IC(c)$ is the Information Content for a class node c, which is calculated as:

$$IC(c) = -logp(c) \tag{7}$$

Here $p(c)$ is the probability of usage of the label in the dataset, which is calculated as the frequency of the label in the dataset. Let us consider the example from the weighted Euclidean and Jaccard distance sub-sections. The SimGIC distance for the two examples (x_1, S_1) and (x_2, S_2) will be:

$$d(S_1, S_2) = 1 - \frac{-log(P(all)P(D)P(D.2)P(D.3))}{-log(P(all)P(B)P(B.1)P(C)P(D)P(D.2)P(D.3))} \tag{8}$$

The ImageCLEF distance is derived from the evaluation score of the ImageCLEF annotation task [9]. This distance can be calculated using the following formula:

$$d_{ImageCLEF}(v_1, v_2) = 1 - \frac{\sum_{c \in labels(v_1) \cap labels(v_2)} \frac{1}{siblings(c)+1} \frac{1}{depth(c)}}{\sum_{c \in labels(v_1) \cup labels(v_2)} \frac{1}{siblings(c)+1} \frac{1}{depth(c)}} \tag{9}$$

where $siblings(c)$ denotes the number of siblings of the class node c in the hierarchy and $depth(c)$ is the depth of the class node c (the root node is omitted in the calculations).

Let us consider the same example as for the weighted Euclidean distance. The ImageCLEF distance for the two examples (x_1, S_1) and (x_2, S_2) will be:

$$d(S_1, S_2) = 1 - \frac{\frac{1}{4}\frac{1}{1} + \frac{1}{3}\frac{1}{2} + \frac{1}{3}\frac{1}{2}}{\frac{1}{4}\frac{1}{1} + \frac{1}{1}\frac{1}{2} + \frac{1}{4}\frac{1}{1} + \frac{1}{4}\frac{1}{1} + \frac{1}{3}\frac{1}{2} + \frac{1}{3}\frac{1}{2}} = \frac{12}{19} \tag{10}$$

2.5 Adaptations of the distance measures for DAGs

The variance (equation 1) is computed using the distance between the class vectors, where a class c's weight $w(c)$ depends on its depth in the class hierarchy (e.g., $w(c) = w_0^{depth(c)}$ with $0 < w_0 < 1$). When the hierarchy structures the classes in the form of a directed acyclic graph (DAG), the depth of a class is not unique since it can have more than one path to a top-level class. An approach was chosen with rewriting the equation $w(c) = w_0^{depth(c)}$ to its recurrent form $w(c) = w_0 w(par(c))$, where $par(c)$ is the parent class of c. Using the equation in this form along with an aggregation function (like sum, min, max, average) several alternatives are possible. In this work we chose to use the average as aggregation function, as recommended in [2]. So, the weighting scheme for DAGs can be defined as follows:

$$w(c) = w_0 avg_j w(par_j(c)) \tag{11}$$

3 Data Description

In this section, we describe the datasets that we used to evaluate the distance measures. We used sixteen datasets from the domain of functional genomics. The datasets represent different aspects of the genes in the genome of Saccharomyces Cerevisiae (12 of the datasets) and Arabidopsis Thaliana (4 of the datasets).

We consider two annotation schemes: FunCat [14] which is a tree-structured class hierarchy and the Gene Ontology (GO) [15], which forms a hierarchy using a directed acyclic graph: each term can have multiple parents (to be more precise, GO's "is-a" relationship between terms is used here).

The basic properties of the datasets are presented in Table 1. The number of examples in each dataset ranges from 1592 to 11763, the number of attributes from 27 to 19628, and the number of nodes in the hierarchy from 250 to 4125.

The datasets include different types of bioinformatic data. The 'pheno' dataset contains information about the phenotype; 'church' and 'eisen' contain data about the expression levels as measured with microarray chips. The 'scop' dataset contains the predicted SCOP class, while 'struc' has the predicted secondary structure. The protein pattern annotations are available in the 'interpro' datasets. Datasets 'spo', 'cellcycle', 'derisi', 'gasch1', 'gasch2' contain microarray data - expression levels of genes of the yeast genome. A more detailed description of the datasets can be found in [10, 2].

4 Experimental design

4.1 Evaluation measures

To measure the predictive performance of the algorithm with the different distance measures we will use Precision-Recall (PR) curves. These curves are obtained by plotting the precision and recall using different thresholds for the obtained probability vectors from the PCTs. Precision is the proportion of positive predictions that are correct, and recall is the proportion of positive examples that are correctly predicted positive. That is,

$$Prec = \frac{TP}{TP + FP} \qquad Prec = \frac{TP}{TP + FN} \tag{12}$$

with TP the number of true positives (correctly predicted positive examples), FP the number of false positives (positive predictions that are incorrect), and FN the number of false negatives (positive examples that are incorrectly predicted negative). Note that these measures ignore the number of correctly predicted negative examples.

Table 1. Basic properties of the used datasets.

	Dataset	Annotation	Number of instances	Number of discrete attributes	Number of continuous attributes	Number of nodes in hierarchy
Saccharomyces Cerevisiae	cellcycle	FunCat	3766	0	77	499
	church	FunCat	3764	1	26	499
	derisi	FunCat	3733	0	63	499
	eisen	FunCat	2425	0	79	461
	gasch1	FunCat	3733	0	173	499
	gasch2	FunCat	3788	0	52	499
	pheno	FunCat	1592	69	0	455
	spo	FunCat	3711	3	77	499
	struc	FunCat	3851	0	19628	499
	church	Gene Ontology	3764	1	26	4125
	eisen	Gene Ontology	2425	0	79	4125
	pheno	Gene Ontology	1592	69	0	3127
Arabidopsis Thaliana	interpro	FunCat	3719	2815	0	263
	scop	FunCat	3097	0	2003	250
	struc	FunCat	3719	14804	0	263
	interpro	Gene Ontology	11763	2815	0	629

The reason why Precision-Recall based evaluation is chosen in this context instead of the ROC analysis, which is more popular, was the following. In functional genomics datasets similar to the ones described and used here, typically only a few genes have been annotated to have a particular function (a particular class in the class hierarchy). This implies that one has to deal with a strongly skewed class distribution were the number of negative instances by far exceeds the number of positive ones [2]. There is a strong interest for correctly predicting the positive instances (that an instance has a given label), rather than the negative ones. ROC curves can present an overly optimistic view of the algorithm's performance (giving rise to a low false positive rate).

We use two approaches to calculate the AUPRC: area under the average PR curve and average area under the PR curve. The first approach uses averages of the precision and recall over all classes, thus obtaining a single curve ($AU(P\bar{R}C)$). The second approach constructs PR curve for each class, and returns the average area under the PR curves for all classes ($AU\bar{P}RC$). The two curves are able to catch different aspects of the performance of the distance measures. The first curve measure uses the information about the frequencies of the classes and the more frequent classes have bigger influence to the final score. On the other hand, the second measure is averaging the performance of each of the classes, i.e. each class has equal contribution to the final score.

4.2 Experimental methodology

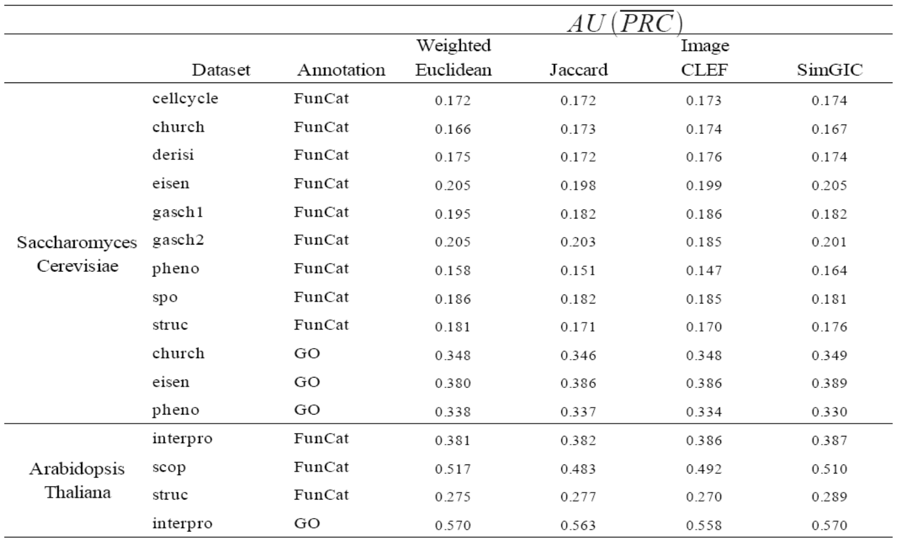
The evaluation of the predictive performance was done using separate testing sets. The threshold value ranged from 0.0 to 1.0 step 0.05. The weight of the depth (w0) was set to 0.75, same as in [2]. Vens et al. in [2] conclude that the weighting parameter has no strong effect on the performance (as compared to non-weighted it gives slightly better results when using Euclidean distance).

To prevent over-fitting, we used two pre-pruning methods: minimal number of examples in a leaf and F-test pruning.The minimal number of examples in a leaf is used as a stopping criterion in the PCT induction algorithm. In our experiments we set this value to 5 examples. The F-test pruning uses the F-test for statistical significance. The F-test is used by the algorithm to check whether the variance reduction is statistically significant at a given significance level. The algorithm takes as input a vector of significance levels and, by internal 10-fold cross-validation it selects one. In our experiments the used vector of significance levels was [0.001, 0.005, 0.01, 0.05, 0.1, 0.125].

5 Results

The performance results of the four different distance measures on the sixteen datasets are summarized in Table 2 and Table 3. As stated in the experimental design section, to evaluate the predictive performance we use the following two measures: the area under the average precision-recall curve and the average area under the PR curves. To check whether the difference in the performance using each of the four distances is statistically significant we used the corrected Friedman test (as recommended in [16]). The corrected Friedman test didn't detect any statistically significant differences in the performance in both cases (p ¡ 0.073 for the are under the average PR curve, and p ¡ 0.176 for the average area under the PR curves).

Table 2. Predictive performance of the algorithms estimated by the area under the average PR curve.



			$AU(\overline{PRC})$			
	Dataset	Annotation	Weighted Euclidean	Jaccard	Image CLEF	SimGIC
Saccharomyces Cerevisiae	cellcycle	FunCat	0.172	0.172	0.173	0.174
	church	FunCat	0.166	0.173	0.174	0.167
	derisi	FunCat	0.175	0.172	0.176	0.174
	eisen	FunCat	0.205	0.198	0.199	0.205
	gasch1	FunCat	0.195	0.182	0.186	0.182
	gasch2	FunCat	0.205	0.203	0.185	0.201
	pheno	FunCat	0.158	0.151	0.147	0.164
	spo	FunCat	0.186	0.182	0.185	0.181
	struc	FunCat	0.181	0.171	0.170	0.176
	church	GO	0.348	0.346	0.348	0.349
	eisen	GO	0.380	0.386	0.386	0.389
	pheno	GO	0.338	0.337	0.334	0.330
Arabidopsis Thaliana	interpro	FunCat	0.381	0.382	0.386	0.387
	scop	FunCat	0.517	0.483	0.492	0.510
	struc	FunCat	0.275	0.277	0.270	0.289
	interpro	GO	0.570	0.563	0.558	0.570

The ranking of the distances by the area under the average PR curve is as follows: the SimGIC distance has the best average rank, followed by the weighted Euclidean distance and the ImageCLEF distance. The Jaccard distance has the worst average rank. The situation is a bit different when average area under the PR curves is used for comparison: the weighted Euclidean distance has the best rank, followed by the SimGIC distance. Next are the ImageCLEF and Jaccard distance with equal rank.

Table 3. Predictive performance of the algorithms estimated by the average area under the PR curves.

			$\overline{AUPRC}$			
	Dataset	Annotation	Weighted Euclidean	Jaccard	Image CLEF	SimGIC
Saccharomyces Cerevisiae	cellcycle	FunCat	0.032	0.027	0.028	0.032
	church	FunCat	0.029	0.028	0.027	0.026
	derisi	FunCat	0.027	0.026	0.028	0.026
	eisen	FunCat	0.047	0.042	0.038	0.040
	gasch1	FunCat	0.036	0.040	0.030	0.038
	gasch2	FunCat	0.034	0.028	0.030	0.029
	pheno	FunCat	0.030	0.030	0.032	0.031
	spo	FunCat	0.030	0.031	0.030	0.032
	struc	FunCat	0.030	0.030	0.027	0.028
	church	GO	0.014	0.015	0.015	0.014
	eisen	GO	0.030	0.022	0.023	0.026
	pheno	GO	0.019	0.017	0.016	0.018
Arabidopsis Thaliana	interpro	FunCat	0.096	0.096	0.099	0.092
	scop	FunCat	0.163	0.139	0.150	0.159
	struc	FunCat	0.045	0.052	0.043	0.052
	interpro	GO	0.139	0.133	0.128	0.142

In Figure 3, we present the average PR curves obtained using the four distances and the 'pheno' dataset with FunCat annotation (Saccharomyces Cerevisiae). We can see that the PR curve for SimGIC is always above the PR-curves for the other distances. It thus clearly performs better than the other distances on this dataset.

6 Conclusions

In this work, we have reviewed and evaluated several distance measures that can be applied in the hierarchical multi-label classification task. In particular, we compared the weighted Euclidean distance, Jaccard distance, SimGIC distance and ImageCLEF distance. The distances were appropriate for hierarchies in the form of a tree, as well as hierarchies in the form of a directed acyclic graph.

We used separate testing sets to evaluate the influence of each distance measure on the learning process. The predictive performance was estimated with the area under the

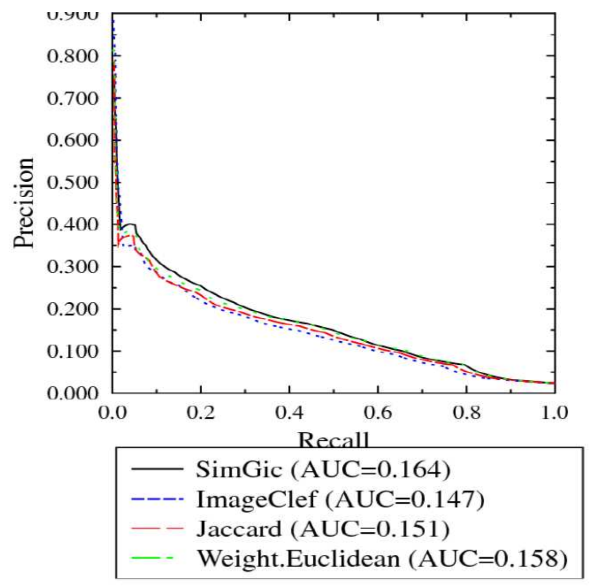


Fig. 3. An example PCT for HMLC, obtained with a given threshold, for the 'church' dataset with FunCat annotation.

average PR curve and the average area under the PR curves. The corrected Friedman test for statistical significance testing didn't detect difference in the performance. However, the SimGIC distance has the best average rank for the area under the average PR curve, while weighted Euclidean distance for the average area under the PR curves.

For future work, we plan to investigate the different weighting schemes. A distance can achieve better predictive performance if used with an appropriate weighting scheme. Also, we will conduct series of experiments on additional datasets from functional genomics and other domains, such as image annotation, text categorization etc.

Another line of further work is the use of ensembles form PCTs [17] to check whether the ensembles can increase the predictive performance and which distance is most suitable for ensemble learning.

Also we plan to investigate other evaluation measures of predictive performance adapted for HMLC [18], such as the hierarchical F-measure, hierarchical Precision, hierarchical Recall, average category similarity and other.

References

1. Barutcuoglu, Z., Schapire, R., Troyanskaya, O.: Hierarchical multi-label prediction of gene function. Bioinformatics **22** (2006) 830–836
2. Vens, C., Struyf, J., Schietgat, L., Dzeroski, S., Blockeel, H.: Decision trees for hierarchical multi-label classification. Machine Learning **73** (2008) 185–214

3. Rousu, J., Saunders, C., Szdemak, S., Shawe-Taylor, J.: Learning hierarchical multi-category text classification models. In: Proc. of the 22nd Int. Conf. on Machine Learning, Omnipress (2005) 745–752
4. Dimitrovski, I. Kocev, D., Loskovska, S., Džeroski, S.: Hierchical annotation of medical images. In: Proc. of the 11th International Multi-Conference Information Society IS 2008, 13. do 17. oktober 2008, Institut "Jozef Stefan", 2008, p. 174-181
5. Bakir, G., Hofmann, T., Scholkopf, B., Smola, A., Taskar, B., Vishwanathan, S. Predicting Structured Data. MIT Press, 2007
6. Blockeel, H., De Raedt, L., Ramon, J. Top-down induction of clustering trees. In: Proc. of the 15th ICML. (1998) 55-63
7. Zenko, B. Learning Predictive Clustering Rules, PhD Thesis, Faculty of Computer and Information Science, University of Ljubljana, 2007
8. Pesquita, C., Faria, D., Bastos, H., Falco, A.O., Couto, F.M. Evaluating GO-based semantic similarity measures In: Proceedings of the 10th Annual Bio-Ontologies Meeting (Bio-Ontologies 2007)
9. Image 2008 Medical Automatic Image Annotation Task http://www.imageclef.org/2008/medaat
10. A. Clare and R. D. King. Knowledge discovery in multi-label phenotype data. *Lecture Notes in Computer Science*, 2168:42–53, 2001.
11. Struyf, J., Džeroski, S. Constraint based induction of multi-objective regression trees, In: Knowledge Discovery in Inductive Databases, 4th International Workshop, KDID'05, LNCS vol. 3933, pp. 222-233, 2006
12. Dzeroski, S., Gjorgjioski, V., Slavkov, I., Struyf, J. Analysis of Time Series Data with Predictive Clustering Trees, In: KDID06, LNCS vol. 4747, p. 63-80, 2007
13. Guo, X., Liu, R., Shriver, C.D., Hu, H. and Liebman, M.N. Assessing semantic similarity measures for the characterization of human regulatory pathways Bioinformatics, 22, 967-973
14. Mewes, H. W., Heumann, K., Kaps, A., Mayer, K., Pfeiffer, F., Stocker, S., Frishman, D. MIPS: a database for protein sequences and complete genomes. Nucl. Acids Research, 27, 1999, 44-48.
15. Ashburner, M. et al. Gene Ontology: tool for the unification of biology. The Gene Ontology Consortium. Nature Genetics, 25(1), 2000, 25-29.
16. Demsar J. Statistical Comparisons of Classifiers over Multiple Data Sets. Journal of Machine Learning Research 7, 2006, 1-30
17. D.Kocev, C. Vens, J. Struyf, S. Džeroski. Ensembles of Multi-Objective Decision Trees, In: Proc. of the ECML 2007, LNAI vol. 4701, p. 624-631, 2007
18. Costa, E.P., Lorena, A.C., Carvalho, A.C.P.L.F., Freitas, A.A. A review of performance evaluation measures for hierarchical classifiers. In: Evaluation Methods for Machine Learning II: papers from the AAAI-2007 Workshop

Combining Classifiers for Improved Multilabel Image Classification

Martin Antenreiter, Ronald Ortner, and Peter Auer

University of Leoben, A-8700 Leoben, Austria
{mantenreit,rortner,auer}@unileoben.ac.at

Abstract. We propose a stacking-like method for multilabel image classification. Our approach combines the output of binary base learners, which use different features for image description, in a simple and straightforward way: The confidence scores of the base learners are fed into a support vector machine (SVM) in order to improve prediction accuracy. Experiments on the datasets of the Pascal Visual Object Classes challenges (VOC) of 2006 and 2007 show that our method significantly improves over the performance of the base learners. Our approach also works better than more naive approaches for combining features or classifiers.

1 Introduction

The idea of combining classifiers in order to get improved prediction accuracy has been considered by many researchers (see e.g. [6, 17] for an overview and further references) and has sparked the development of seminal methods such as *boosting* [11, 26]. Here we consider *multilabel* learning problems where each instance may have several labels (unlike in *multiclass* problems where each instance is assigned to a unique class, i.e., has a single label). We present a very simple method for multilabel learning based on the combination of binary base classifiers. That is, we propose to train for each label a binary base classifier. Then we feed the output (i.e., the confidence scores) of these binary base learners into a support vector machine (SVM) in order to improve prediction accuracy.

The basic concept underlying this simple approach resembles *stacking* [28] methods: In the stacking framework the output of several (distinct) base classifiers is combined by a meta-level learner to give an improvement in (binary or multiclass) classification. In the multiclass case usually multiclass classifiers are used as base learners and as meta-level learners (cf. [8]). While the idea behind stacking is that the metalearner shall be able to combine the base learners in a more sophisticated way than doing simple voting or cross-validation [28], in our method the meta-level learner shall grasp interdependencies between the single classes that the base learners have not properly captured.

There has been some discussion whether stacking really gives any improvement over choosing the best base classifier [8] (see also [25] for a related discussion). In any case, it is known that the success of stacking methods depends on the choice of the meta-level learner as well as the kind of input this learner takes from the base learners [27].

Recent suggestions (see [8] for an overview) usually use the probability distributions predicted by the base learners (in some form) as input for the meta-level learner. Similarly, we use the base learners' confidence scores, which is usually simpler than working with probability distributions, as obtaining the latter requires additional optimization methods [23]. The choice for the meta-level learners in stacking approaches ranges from nearest neighbor [20] to tree methods [8]. SVMs have been used as metalearners as well [1, 7, 18].

We tested our algorithm on the popular image classification databases, provided for the VOC challenges in 2006 and 2007 [10, 9]. Instead of combining essentially different base learners, we kept the base learning algorithm fixed, while using different features for describing the image data. Results show that the combined classifiers outperform the base classifiers in every experiment, which indicates that the combined classifiers are indeed able to extract interdependencies between the individual classes and also the individual classifiers.

The rest of the paper is organized as follows. The following section describes our algorithm in more detail and discusses some related work. Section 3 describes the datasets and the experimental setup as well as the results. There, we also consider alternative methods for combining features or classifiers to which we compare our approach. The final section considers directions of future work.

2 Multilabel Classification

2.1 General Considerations

In multilabel classification problems there are usually interdependencies between classes. For example, when labeling images it is rather unlikely that an image containing a sheep also shows an aeroplane. On the other hand, images containing cars will usually also contain roads. The art of multilabel classification lies in reliably detecting and exploiting these interdependencies. A base classifier that is trained on enough examples will also learn these interdependencies. However, in the common case where the training set is not sufficiently large, a base learner for cars may not fully grasp the interdependency between cars and roads. This defect can be corrected by having a base learner for roads that will be able to learn roads from more training examples than just those where also cars appear. That way, combining the road classifier with the car classifier in a suitable way will also improve the performance for classification of cars.

Our algorithm uses binary base classifiers that are trained to recognize single classes. In order to better capture interdependencies between the individual classes we propose combining the confidence scores of the different base classifiers by simply feeding them into a support vector machine (SVM).

2.2 The Basic Algorithm

We consider the following *multilabel* problem: Given is a set of training examples $\{x_1, \ldots, x_n\} \subset X$ together with labels for N different classes $\{C_1, \ldots, C_N\}$ (where each $C_k \subseteq X$). That is, for each training instance x_i and each class C_k the respective label is

$$y_i^{(k)} := \begin{cases} +1 & \text{if } x_i \in C_k \\ -1 & \text{otherwise.} \end{cases}$$

As the problem is assumed to be *multilabel* but not necessarily *multiclass*, $y_i^{(k)} = 1$ not necessarily implies that $y_i^{(\ell)} = 0$ for $\ell \neq k$. Thus, the classes C_k in general will not partition the instance space X.

We first train for each class C_k a corresponding (binary) base learner $h^{(k)}$ that shall be able to predict the labels $y_i^{(k)}$ well. More generally, when using M distinct classification algorithms for each single class C_k, we have a total of $N \cdot M$ base classifiers. Each base classifier returns a confidence score s for each training example x_i. In our case this will be a real value (the distance to the separating hyperplane) that is positive if the classifier predicts that x_i is in the target class and negative otherwise. However, more generally this score could also be a real number with a different interpretation (e.g. a probability distribution as in recent stacking approaches). Let $s_{jk}(x)$ be the confidence score returned by base classifier $h_j^{(k)}$ for class C_k on training instance x. The collected confidence scores are then combined by N metalearners, one for each class C_k, where the training set consists of the vectors

$$\mathbf{v}_i = \big(s_{1,1}(x_i), s_{1,2}(x_i), ..., s_{1,N}(x_i), s_{2,1}(x_i), ..., s_{2,N}(x_i), ..., s_{M,N}(x_i)\big) \quad (1)$$

for all training instances x_i. The label of each $\mathbf{v}_i$ is simply the label $y_i^{(k)}$ of x_i with respect to class C_k.

2.3 Algorithm Specification for Image Classification

A state-of-the-art approach for image classification is the following: First, features are extracted from the images. These features then are clustered to generate a visual codebook. Based on that codebook a histogram of "visual words" for every image is built. This approach was inspired by the text classification community where it is called "bag-of-words". In text classification the input features are words, while in image classification descriptors take over this part. A powerful descriptor is the scale-invariant feature transform (SIFT) [19], although there are other very good descriptors for certain image classes. A common approach is to build for every descriptor type a histogram and concatenate these histograms.

Our base classifiers work with different features (as specified in Section 3 below) that are learned with a fixed learning algorithm (in our case either LPBoost or Fisher kernels) to give the confidence scores. As metalearner SVMs [4] are deployed.

2.4 Some Related Work

There is a lot of work dealing with multilabel problems in far more complex ways than our straightforward approach. Thus, in [18] SVMs are used for combining different types of features for mapping of proteins to Gene Ontology. Their method trains

$\frac{N\cdot(N-1)}{2}$ binary classifiers for a problem with N labels. A function for combining and normalizing the output of the $\frac{N\cdot(N-1)}{2}$ binary classifiers is used and the normalization parameters have to be estimated. Additional knowledge of the problem domain is integrated by a directed acyclic graph to speed up the final classification.

Actually, it is quite common to model and use some additional context information in order to process the output of the base learners. This work is subsumed under the term of *Context Based Concept Fusion* (CBCF). For some references and an integrated approach see e.g. [24].

A simpler approach that has more in common with our method has been suggested in [13]. There it is proposed to use the base learners' output labels as additional coordinates in the training vectors. However, unlike our algorithm this does not consider the confidence of the base learners.

As already mentioned in the introduction, there is also a lot of work using classical stacking approaches for classifier combination that resembles our method. See e.g. the recent [1] which suggests stacking with SVMs in a multiclass image classification problem.

Compared to these exemplary alternative approaches, we find that our method is appealingly simple, and — as will be seen — works surprisingly well.

3 Experiments

3.1 Other Ways of Feature/Classifier Combination

In the experiments, we compared our approach with other ways to combine the base classifiers.

Using All Features As a baseline on the first dataset (VOC 2006) we compare our approach to the more direct combination of the features by jointly using them for training the base classifiers. More precisely, the base learner — the boosting algorithm LPBoost — may choose in each boosting iteration the best single feature type for a decision stump. We used the boosting approach with decision stumps, because it is very suitable for combining different kinds of feature types.

Binary Stacking In order to show that our algorithm profits from the confidence information of the other classes, we also did a comparison to the following alternative method where the metalearner uses for learning class C_k not the whole vectors $\mathbf{v}_i$ as given in (1) but only the confidence information for the class C_k at question. That is, the training vectors in this case are

$$\mathbf{v}_i = \big(s_{1,k}(x_i), ..., s_{M,k}(x_i)\big)$$

for each training instance x_i with label $y_i^{(k)}$. This corresponds to classical stacking on a binary classification problem, and we call this method *binary stacking* in what follows. Indeed, binary stacking with some minor modifications has been considered and empirically evaluated (among other multilabel algorithms) in [7].

The Best Binary Base Classifier Finally, we do a challenging comparison of our approach to the best binary base classifier. That is, we choose for the prediction of each class C_k the base classifier h_i that gives the best prediction accuracy on the test examples. Note that this classifier is usually unknown beforehand, so that choosing this best binary base classifier has an advantage over our method. However, even in this setting we show that our method gives better results.

3.2 Data Sets and Setup

We conducted experiments on the well-known image classification databases taken from the Pascal Visual Object Classes Challenges 2006 (VOC 2006) [10] and 2007 (VOC 2007) [9]. The VOC 2006 dataset contains 10 classes in 5,304 images, on which a total of 9,507 annotated objects can be found. The VOC 2007 dataset contains 20 classes in 9,963 images with 24,640 annotated objects. Both are multilabel datasets. These datasets are split into a fixed training, validation, and test set. In the VOC 2006 database the training set contains 1,277 images. The validation set has 1,341 images and the test set 2,686 images. The VOC2007 database has 2,501 training images, 2,510 validation images, and 4,952 test images.

The training of the base learners was done on the training dataset using the independent validation set for parameter selection. The selection of the kernel and the parameters was done using 5-fold cross-validation on the validation data. After that, we learned the combined classifiers using the validation data. All reported results are from the evaluation on the test data. As evaluation criterion we use the average precision as for the original VOC challenges.

3.3 Experiments on the VOC 2006 Dataset

In the experiments on the VOC 2006 dataset, we trained for each of the ten classes a classifier using LPBoost [3, 5] together with an image descriptor taken from a set of nine different feature types. The first feature type uses texture statistics of segment regions from a segmentation algorithm [12]. All other feature types are calculated from regions of interest obtained from a scale invariant Harris-Laplace detector [21]. On those regions subsampled grayvalues, basic moments, moment invariants [14], SIFT [19], and PCA-SIFT [16] descriptors are calculated. Further, we obtain three additional feature types by intensity normalization of the subsampled grayvalue, of the basic moments, and of the moment invariants descriptors. An overview over the used feature types is given in Tab. 1.

For our algorithm we used each feature type together with LPBoost to obtain for each class a total of nine binary classifiers as base learners. The output (the distance to the separating hyperplane) of these base learners (for all classes) was then fed into an SVM metalearner as described in Section 2.2, see (1). For completeness we tested our approach with different kernels. However, results show that the choice of the SVM kernel function is not critical.

We compared our approach to the best of the base classifiers. An overview of the performance of each descriptor on the ten classes can be found in Tab. 2. It can be seen from Tab. 3 that our combined classifier outperforms the individual classifiers, even if

h_1 ...	texture statistics of segments
h_2 ...	subsampled grayvalues
h_3 ...	subsampled grayvalues (intensity normalized)
h_4 ...	basic moments
h_5 ...	basic moments (intensity normalized)
h_6 ...	moment invariants
h_7 ...	moment invariants (intensity normalized)
h_8 ...	SIFTs
h_9 ...	PCA-SIFTs

Table 1. Feature types for the experiments on the VOC 2006 dataset.

we choose for each class that base classifier that gives the best performance on the test set.

As already indicated before, another comparison was made to the algorithm where the weak learner of LPBoost may choose in each boosting iteration one reference feature among the nine different feature types. Thus, in this setting the boosting algorithm is not restricted to a single feature type and may choose the best feature with the optimal threshold. This approach (denoted 'original classifier $h_{1..9}$' in Tab. 3 and 1) has been used in the VOC 2006 Challenge and is used as a base line for our experiments. For more details see [2]. While the results for this alternative algorithm sometimes improve even over the best single classifier, it is still outperformed by our algorithm. The collected results can be found in Tab. 3.

Fig. 1 shows a comparison of our method with the binary stacking approach. For binary stacking we used the same nine base classifiers that are combined by an SVM. We report only the results of binary stacking with an RBF-kernel SVM as metalearner, which performed best. The resulting performance of binary stacking is on some classes slightly better than our base line where all features are used from the beginning. How-

class	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9
bicycle	17.45	40.94	40.56	41.75	39.40	37.94	31.17	**56.84**	54.73
bus	9.57	25.94	**30.17**	25.38	19.39	13.60	16.30	16.03	25.62
car	40.78	**70.72**	69.99	48.74	51.33	38.17	41.41	56.90	60.47
cat	16.23	**29.60**	28.69	15.49	15.30	14.53	14.89	15.68	15.13
cow	8.99	18.15	15.40	17.62	18.74	10.18	17.48	17.64	**26.14**
dog	14.08	20.06	20.22	18.12	14.36	15.92	15.43	**22.86**	15.69
horse	9.94	12.29	16.86	**19.58**	13.08	14.65	11.49	10.06	12.24
motorbike	13.22	27.10	29.11	32.96	**39.90**	28.35	26.19	10.54	12.26
person	29.66	39.92	36.94	38.34	**43.47**	36.01	42.55	31.51	31.97
sheep	8.99	24.99	25.34	21.85	19.47	11.06	15.92	29.42	**36.37**
avg	16.89	30.97	31.33	27.98	27.44	22.04	23.28	26.75	29.06

Table 2. Average precision of the individual classifiers on the VOC 2006 dataset in percent. Bold values indicate the best classifier on a given class for the test set.

class	$\max(h_1, \ldots, h_9)$	original $h_{1..9}$	our (linear)	our (polynomial)	our (RBF)
bicycle	56.84	61.12	**61.36**	56.16	60.77
bus	30.17	27.32	**51.66**	50.00	51.54
car	70.72	70.92	74.03	**76.30**	74.83
cat	29.60	24.41	37.13	**41.15**	37.98
cow	**26.14**	18.92	23.41	24.20	25.56
dog	22.86	25.88	32.16	34.95	**36.41**
horse	19.58	12.12	22.44	**27.44**	20.86
motorbike	39.90	33.19	47.09	46.92	**48.35**
person	43.47	35.16	42.55	**46.56**	43.04
sheep	36.37	29.39	**41.87**	37.07	40.55
avg	37.57	33.84	43.37	**44.07**	43.99

Table 3. Comparison of the best individual classifier, the classifier using all descriptor types in the beginning, and our approach on the VOC 2006 dataset. Results are in percentage using the average precision measure. Bold values indicate the optimal method.

ever, on some other classes binary stacking suffers a performance decrease of up to 9.6%. The mean average precision of binary stacking is 30.52%, which compared to our base line means a performance loss of 3.32%. These experiments may also confirm doubts concerning the utility of stacking. However, as already mentioned before, usually class probabilities instead of confidence scores are used for stacking. The choice of the latter may affect the results to the negative. Indeed, a similar stacking method [1] in an image classification problem where class probabilities are the input for the metalearner has been more successful. That confidence scores work fine in our proposed method can be interpreted the way that our metalearner SVMs do the normalization that also has to be done when trying to obtain probability distributions from confidence scores.

3.4 Experiments on the VOC 2007 Dataset

In the experiments on the VOC 2007 database we had only access to two base classifiers. The first classifier h_1 is based on texture information using the SIFT descriptor [19], while the second classifier h_2 is based on Gaussian weighted local color information (h_2). Both descriptors are extracted from a dense grid at five different scales. Each classifier is learned with the Fisher kernels framework [22], which gives state-of-the-art performance on the VOC 2007 database. Tab. 4 shows that the performance of the classifier using the SIFT descriptor yields consistently better results than the descriptor based on the local color information (with the only exception being the class ‘pottedplant’). In spite of this and the fact that the information of two classifiers is quite limited, our method was able to improve the mean of the average precision across all the 20 categories by up to 3.46% (for the RBF kernel). When using cross-validation to choose the kernel, the RBF kernel is selected for all classes except one (cf. Tab. 4) giving the same average precision as for the RBF kernel. The collected results can be found in Tab. 4.

In Tab. 5 we compare our method with binary stacking. In this experiment a linear kernel gave the best results for binary stacking, but it can be seen that our method gives

class	$h_{1..9}$	stacking	our (RBF)
bicycle	**61.12**	56.77	60.77
bus	27.32	27.52	**51.54**
car	70.92	65.86	**74.83**
cat	24.41	14.75	**37.98**
cow	18.92	11.83	**25.56**
dog	25.88	17.55	**36.41**
horse	12.12	12.42	**20.86**
motorbike	33.19	30.21	**48.35**
person	35.16	34.25	**43.04**
sheep	29.39	34.00	**40.55**
avg	33.84	30.52	**43.99**

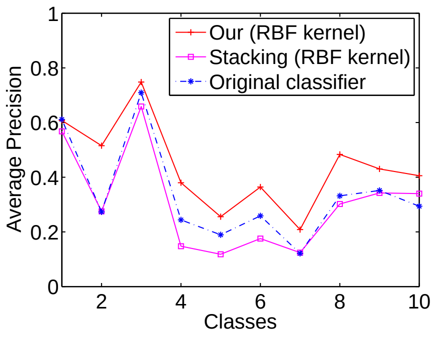


Fig. 1. Comparison of the classifier using all descriptor types in the beginning, binary stacking, and our approach on the VOC 2006 dataset. Results are in percentage using the average precision measure. For binary stacking and our method we report the values obtained for the RBF kernel.

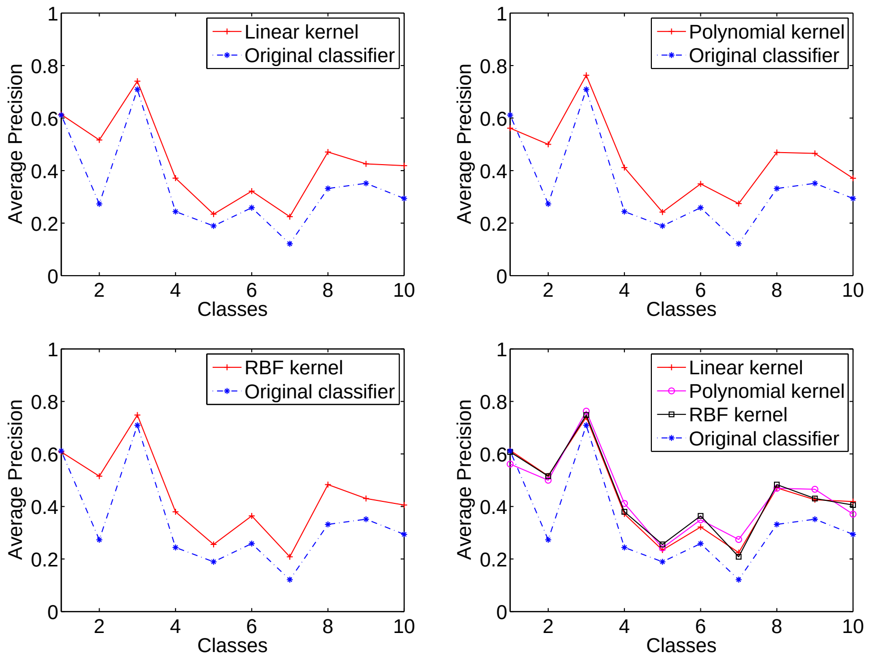


Fig. 2. Comparison of the classifier using all descriptor types in the beginning and our approach using different kernels on the VOC 2006 dataset. Results are in percentage using the average precision measure. The choice of the SVM kernel function can be seen to be not critical.

class	h_1	h_2	our (linear)	impr.	our (poly.)	impr.	our (RBF)	impr.
aeroplane	66.41	59.51	65.88	-0.53	65.47	-0.94	**66.71***	0.30
bicycle	47.31	35.45	51.09	3.78	53.23	5.92	**53.42***	6.11
bird	44.45	42.67	49.99	5.54	53.06	8.61	**53.47***	9.02
boat	58.87	41.12	63.21	4.34	**63.26**	4.39	62.38*	3.51
bottle	24.18	15.16	25.97	1.79	**27.53**	3.35	23.93*	-0.25
bus	**52.42**	34.24	51.65	-0.77	43.64	-8.78	45.75*	-6.67
car	70.70	56.47	70.67	-0.03	**73.32***	2.62	**73.32**	2.62
cat	45.30	39.49	44.87	-0.43	44.78	-0.52	**46.30***	1.00
chair	47.11	37.78	50.68	3.57	49.82	2.71	**50.72***	3.61
cow	31.25	15.03	29.00	-2.25	31.28	0.03	**32.99***	1.74
diningtable	38.21	35.75	42.29	4.08	43.12	4.91	**44.71***	6.50
dog	40.98	33.44	38.77	-2.21	40.42	-0.56	**41.95***	0.97
horse	67.77	64.48	71.44	3.67	73.46	5.69	**73.48***	5.71
motorbike	52.37	46.02	55.07	2.70	56.05	3.68	**57.85***	5.48
person	80.17	78.33	82.43	2.26	83.01	2.84	**83.22***	3.05
pottedplant	24.30	27.14	28.71	1.57	29.11	1.97	**32.92***	5.78
sheep	27.32	25.48	36.47	9.15	28.76	1.44	**38.79***	11.47
sofa	**44.36**	31.57	41.81	-2.55	40.67	-3.69	40.73*	-3.63
train	65.21	54.42	66.88	1.67	69.52	4.31	**69.87***	4.66
tvmonitor	41.34	34.26	44.54	3.20	**48.27**	6.93	46.66*	5.32
avg	48.50	40.39	50.57	2.07	50.89	2.39	**51.96**	3.46

Table 4. Average precision on the VOC 2007 dataset in percent for the two base classifiers as well as for our method with linear, polynomial and RBF kernel. The best method for each class is indicated by a bold entry. Starred values indicate which kernel is chosen by cross-validation.

better results for 14 classes. Our average improvement to the base classifiers is 3.46%, whereas binary stacking only improves by 1.66%.

4 Conclusion

While our presented method of combining classifiers for multilabel classification is appealingly simple, it works quite well. Our main goal was to investigate the gain of our approach with respect to the base learners and simpler methods for combining features or classifiers. However, as higher complexity not necessarily results in improved performance [15, 8], it would of course be interesting to compare our elementary method to more complex algorithms for combination of classifiers. Another direction of future work is to test our approach on similar problems with different features available, such as in text classification.

Acknowledgments The authors would like to thank Teo de Campos from XEROX for providing the confidence scores for the experiments on the VOC 2007 dataset. This work was supported in part by the Austrian Science Fund FWF (S9104-N13 SP4). The

class	h_1	h_2	stacking	impr.	our (RBF)	impr.
aeroplane	66.41	59.51	**68.16**	1.75	66.71	0.30
bicycle	47.31	35.45	48.61	1.30	**53.42**	6.11
bird	44.45	42.67	49.37	4.92	**53.47**	9.02
boat	58.87	41.12	60.65	1.78	**62.38**	3.51
bottle	24.18	15.16	**25.76**	1.58	23.93	-0.25
bus	**52.42**	34.24	51.15	-1.27	45.75	-6.67
car	70.70	56.47	69.72	-0.98	**73.32**	2.62
cat	45.30	39.49	**46.86**	1.56	46.30	1.00
chair	47.11	37.78	45.18	-1.93	**50.72**	3.61
cow	31.25	15.03	30.96	-0.29	**32.99**	1.74
diningtable	38.21	35.75	38.51	0.30	**44.71**	6.50
dog	40.98	33.44	**43.20**	2.22	41.95	0.97
horse	67.77	64.48	70.61	2.84	**73.48**	5.71
motorbike	52.37	46.02	56.45	4.08	**57.85**	5.48
person	80.17	78.33	81.83	1.66	**83.22**	3.05
pottedplant	24.30	27.14	29.50	2.36	**32.92**	5.78
sheep	27.32	25.48	30.97	3.65	**38.79**	11.47
sofa	44.36	31.57	**45.15**	0.79	40.73	-3.63
train	65.21	54.42	67.76	2.55	**69.87**	4.66
tvmonitor	41.34	34.26	42.75	1.41	**46.66**	5.32
avg	48.50	40.39	50.16	1.66	**51.96**	3.46

Table 5. Average precision on the VOC 2007 dataset in percent for the two base classifiers as well as for binary stacking and our method with RBF kernel.

research leading to these results has also received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreements n° 216886 (PASCAL2 Network of Excellence), and n° 216529 (PinView). This publication only reflects the authors' views.

References

1. Azizi Abdullah, Remco Veltkamp, and Marco Wiering. Spatial pyramids and two-layer stacking SVM classifiers for image categorization: A comparative study. In *International Joint Conference on Neural Networks (IJCNN 2009)*.
2. Martin Antenreiter, Christian Savu-Krohn, and Peter Auer. Visual classification of images by learning geometric appearances through boosting. In *Artificial Neural Networks in Pattern Recognition (ANNPR 2006)*, pages 233–243, 2006.
3. Kristin P. Bennett, Ayhan Demiriz, and John Shawe-Taylor. A column generation algorithm for boosting. In *Proceedings of the 17th International Conference on Machine Learning (COLT 2000)*, pages 65–72. Morgan Kaufmann, 2000.
4. Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.
5. Ayhan Demiriz, Kristin P. Bennett, and John Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1-3):225–254, 2002.

6. Thomas G. Dietterich. Machine-learning research: Four current directions. *The AI Magazine*, 18(4):97–136, 1998.
7. Anastasios Dimou, Grigorios Tsoumakas, Vasileios Mezaris, Ioannis Kompatsiaris, and Ioannis Vlahavas. An empirical study of multi-label learning methods for video annotation. In *7th International Workshop on Content-Based Multimedia Indexing (CBMI 2009)*, 2009.
8. Sašo Džeroski and Bernard Ženko. Is combining classifiers with stacking better than selecting the best one? *Machine Learning*, 54(3):255–273, 2004.
9. Mark Everingham, Luc J. Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.
10. Mark Everingham, Andrew Zisserman, Christopher K. I. Williams, and Luc J. Van Gool. The PASCAL Visual Object Classes Challenge 2006 (VOC2006) Results. http://www.pascal-network.org/challenges/VOC/voc2006/results.pdf.
11. Yoav Freund. Boosting a weak learning algorithm by majority. In *Proceedings of the Third Annual Workshop on Computational Learning Theory (COLT 1990)*, pages 202–216, 1990.
12. Michael Fussenegger, Andreas Opelt, Axel Pinz, and Peter Auer. Object recognition using segmentation for feature detection. In *International Conference on Pattern Recognition (ICPR) (3)*, pages 41–44, 2004.
13. Shantanu Godbole and Sunita Sarawagi. Discriminative methods for multi-labeled classification. In *Advances in Knowledge Discovery and Data Mining, 8th Pacific-Asia Conference (PAKDD 2004)*, pages 22–30. Springer, 2004.
14. Luc J. Van Gool, Theo Moons, and Dorin Ungureanu. Affine/photometric invariants for planar intensity patterns. In *ECCV '96: Proceedings of the 4th European Conference on Computer Vision – Volume I*, pages 642–651. Springer, 1996.
15. Robert C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91, 1993.
16. Yan Ke and R. Sukthankar. PCA-SIFT: a more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition (CVPR 2004)*, volume 2, pages 506–513. IEEE Computer Society, 2004.
17. Josef Kittler, Mohamad Hatef, Robert P. W. Duin, and Jiri Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.
18. Hans-Peter Kriegel, Peer Kröger, Alexey Pryakhin, and Matthias Schubert. Using support vector machines for classifying large sets of multi-represented objects. In *Proceedings of the Fourth SIAM International Conference on Data Mining (SDM 2004)*. SIAM, 2004.
19. David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
20. Christopher J. Merz. Using correspondence analysis to combine classifiers. *Machine Learning*, 36(1-2):33–58, 1999.
21. Krystian Mikolajczyk and Cordelia Schmid. Indexing based on scale invariant interest points. In *International Conference on Computer Vision (ICCV 2001)*, pages 525–531, 2001.
22. Florent Perronnin and Christopher Dance. Fisher kernels on visual vocabularies for image categorization. In *Computer Vision and Pattern Recognition (CVPR 2007)*. IEEE Computer Society, 2007.
23. John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In Peter J. Bartlett, Bernhard Schölkopf, Dale Schuurmans, and Alex J. Smola, editors, *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, Boston, 1999.
24. Guo-Jun Qi, Xian-Sheng Hua, Yong Rui, Jinhui Tang, Tao Mei, and Hong-Jiang Zhang. Correlative multi-label video annotation. In *Proceedings of the 15th International Conference on Multimedia 2007 (MM 2007)*, pages 17–26. ACM, 2007.

25. Ryan M. Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.
26. Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.
27. Kai Ming Ting and Ian H. Witten. Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10:271–289, 1999.
28. David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

A Simple Instance-Based Approach to Multilabel Classification Using the Mallows Model

Weiwei Cheng and Eyke Hüllermeier

Department of Mathematics and Computer Science
University of Marburg, Germany
{cheng,eyke}@mathematik.uni-marburg.de

Abstract. Multilabel classification is an extension of conventional classification in which a single instance can be associated with multiple labels. Recent research has shown that, just like for standard classification, instance-based learning algorithms relying on the nearest neighbor estimation principle can be used quite successfully in this context. In this paper, we propose a new instance-based approach to multilabel classification, which is based on calibrated label ranking, a recently proposed framework that unifies multilabel classification and label ranking. Within this framework, instance-based prediction is realized is the form of MAP estimation, assuming a statistical distribution called the Mallows model.

1 Introduction

In conventional classification, each instance is assumed to belong to exactly one among a finite set of candidate classes. As opposed to this, the setting of multilabel classification allows an instance to belong to several classes simultaneously or, say, to attach more than one label to an instance. Multilabel classification has received increasing attention in machine learning in recent years.

Even though quite a number of sophisticated methods for multilabel classification has been proposed in the literature, the application of *instance-based learning* (IBL) has not been studied very deeply in this context so far. This is a bit surprising, given that IBL algorithms based on the nearest neighbor estimation principle have been applied quite successfully in classification and pattern recognition for a long time [1]. A notable exception is the *multilabel k-nearest neighbor* (MLKNN) method that was recently proposed in [2], where it was shown to be competitive to state-of-the-art machine learning methods.

In this paper, we introduce a new instance-based approach to multilabel classification, which is based on calibrated label ranking, a recently proposed framework that unifies multilabel classification and label ranking (see Section 2). Within this framework, instance-based prediction is realized in the form of MAP estimation, assuming a statistical distribution called the Mallows model (see Section 3). Experimental results (presented in Section 5, subsequent to an overview of related work in Section 4) provide evidence for the strong performance of this approach in terms of predictive accuracy.

2 Multilabel Classification as Calibrated Label Ranking

Let $\mathbb{X}$ denote an instance space and let $\mathcal{L} = \{\lambda_1, \lambda_2 \dots \lambda_m\}$ be a finite set of class labels. Moreover, suppose that each instance $\boldsymbol{x} \in \mathbb{X}$ can be associated with a subset of labels $L \in 2^{\mathcal{L}}$; this subset is often called the set of *relevant* labels, while the complement $\mathcal{L} \setminus L$ is considered as *irrelevant* for $\boldsymbol{x}$. Given training data in the form of a finite set T of observations in the form of tuples $(\boldsymbol{x}, L_{\boldsymbol{x}}) \in \mathbb{X} \times 2^{\mathcal{L}}$, typically assumed to be drawn independently from an (unknown) probability distribution on $\mathbb{X} \times 2^{\mathcal{L}}$, the goal in multilabel classification is to learn a classifier $h : \mathbb{X} \rightarrow 2^{\mathcal{L}}$ that generalizes well beyond these observations in the sense of minimizing the expected prediction loss with respect to a specific loss function.

Note that multilabel classification can be reduced to a conventional classification problem in a straightforward way, namely by considering each label subset $L \in 2^{\mathcal{L}}$ as a distinct (meta-)class. This approach is referred to as *label powerset* in the literature. An obvious drawback of this approach is the potentially large number of classes that one has to deal with in the newly generated problem. Another way of reducing multilabel to conventional classification is offered by the *binary relevance* (BR) approach. Here, a single binary classifier h_i is trained for each label $\lambda_i \in \mathcal{L}$. For a query instance $\boldsymbol{x}$, this classifier is supposed to predict whether λ_i is relevant for $\boldsymbol{x}$ ($h_i(\boldsymbol{x}) = 1$) or not ($h_i(\boldsymbol{x}) = 0$). A multilabel prediction for $\boldsymbol{x}$ is then given by $h(\boldsymbol{x}) = \{\lambda_i \in \mathcal{L} \,|\, h_i(\boldsymbol{x}) = 1\}$. Since binary relevance learning treats every label independently of all other labels, an obvious disadvantage of this approach is that it ignores potential correlations and interdependencies between labels.

Some of the more sophisticated approaches learn a multilabel classifier h in an indirect way via a scoring function $f : \mathbb{X} \times \mathcal{L} \rightarrow \mathbb{R}$ that assigns a real number to each instance/label combination. Such a function does not only allow one to make multilabel predictions (via thresholding the scores), but also offers the possibility to produce a ranking of the class labels, simply by ordering them according to their score. Sometimes, this ranking is even more desirable as a prediction, and indeed, there are several evaluation metrics that compare a true label subset with a predicted ranking instead of a predicted label subset.

In the following, we propose a formalization of multilabel classification within the framework of label ranking. More specifically, as will be seen, this framework allows one to combine the concepts of a ranking and a multilabel prediction (label subset) in a convenient way.

2.1 Label Ranking

The problem of *label ranking*, which has recently been introduced in machine learning [3, 4], can be seen as another extension of the conventional classification setting. Instead of associating every instance $\boldsymbol{x} \in \mathbb{X}$ with one among a finite set of class labels $\mathcal{L} = \{\lambda_1, \lambda_2 \dots \lambda_m\}$, we associate $\boldsymbol{x}$ with a total order of all class labels, that is, a complete, transitive, and asymmetric relation $\succ_{\boldsymbol{x}}$ on $\mathcal{L}$, where $\lambda_i \succ_{\boldsymbol{x}} \lambda_j$ indicates that λ_i precedes λ_j. Since a ranking can be considered as a special type of preference relation, we shall also say that $\lambda_i \succ_{\boldsymbol{x}} \lambda_j$ indicates that λ_i is *preferred* to λ_j given the instance $\boldsymbol{x}$.

Formally, a total order $\succ_{\boldsymbol{x}}$ can be identified with a permutation $\pi_{\boldsymbol{x}}$ of the set $\{1 \dots m\}$. It is convenient to define $\pi_{\boldsymbol{x}}$ such that $\pi_{\boldsymbol{x}}(i) = \pi_{\boldsymbol{x}}(\lambda_i)$ is the position of λ_i in the order. This permutation encodes the (ground truth) order relation

$$\lambda_{\pi_{\boldsymbol{x}}^{-1}(1)} \succ_{\boldsymbol{x}} \lambda_{\pi_{\boldsymbol{x}}^{-1}(2)} \succ_{\boldsymbol{x}} \dots \succ_{\boldsymbol{x}} \lambda_{\pi_{\boldsymbol{x}}^{-1}(m)} \ ,$$

where $\pi_{\boldsymbol{x}}^{-1}(j)$ is the index of the label put at position j. The class of permutations of $\{1 \dots m\}$ (the symmetric group of order m) is denoted by Ω. By abuse of terminology, though justified in light of the above one-to-one correspondence, we refer to elements $\pi \in \Omega$ as both permutations and rankings.

In analogy with the classification setting, we do not assume the existence of a deterministic $\mathbb{X} \to \Omega$ mapping. Instead, every instance is associated with a *probability distribution* over Ω. This means that, for each $\boldsymbol{x} \in \mathbb{X}$, there exists a probability distribution $\mathbf{P}(\cdot \,|\, \boldsymbol{x})$ such that, for every $\pi \in \Omega$, $\mathbf{P}(\pi \,|\, \boldsymbol{x})$ is the probability that $\pi_{\boldsymbol{x}} = \pi$.

The goal in label ranking is to learn a "label ranker" in the form of an $\mathbb{X} \to \Omega$ mapping. As training data, a label ranker uses a set of instances $\boldsymbol{x}_k$, $k = 1 \dots n$, together with information about one or more pairwise preferences of the form $\lambda_i \succ_{\boldsymbol{x}_k} \lambda_j$. To evaluate the predictive performance of a label ranker, a suitable loss function on Ω is needed. In the statistical literature, several distance measures for rankings have been proposed. One commonly used measure is the number of discordant label pairs,

$$D(\pi, \sigma) \ = \ \#\{(i,j) \,|\, \pi(i) > \pi(j) \ \wedge \sigma(i) < \sigma(j)\} \, , \qquad (1)$$

which is closely related to Kendall's tau coefficient. In fact, the latter is a normalization of (1) to the interval $[-1, +1]$. We shall focus on Kendall's tau as a natural, intuitive, and easily interpretable measure [5] throughout the paper, even though other distance measures could of course be used. A desirable property of any distance $D(\cdot)$ is its invariance toward a renumbering of the elements (renaming of labels). This property is equivalent to the *right invariance* of $D(\cdot)$, namely $D(\sigma\nu, \pi\nu) = D(\sigma, \pi)$ for all $\sigma, \pi, \nu \in \Omega$, where $\sigma\nu = \sigma \circ \nu$ denotes the permutation $i \mapsto \sigma(\nu(i))$. The distance (1) is right-invariant, and so are most other commonly used metrics on Ω.

2.2 Calibrated Label Ranking

A label ranking provides information about the *relative* preference for labels, but not about the absolute preference or, say, relevance of a label. To combine the information offered by a label ranking and a multilabel classification (label subset), the concept of a *calibrated label ranking* has been proposed in [6]. A calibrated label ranking is a ranking of the label set Ω extended by a *neutral label* λ_0. The idea is that λ_0 splits a ranking into two parts, the positive (relevant) part consisting of those labels λ_i preceding λ_0 (i.e., $\lambda_i \succ_{\boldsymbol{x}} \lambda_0$), and the negative (irrelevant) part given by those labels λ_j ranked lower than λ_0 (i.e., $\lambda_0 \succ_{\boldsymbol{x}} \lambda_j$). In this way, a multilabel prediction can be derived from a (predicted) calibrated label ranking.

The other way around, a multilabel set $L_{\boldsymbol{x}}$ translates into the set of pairwise preferences $\{\lambda \succ_{\boldsymbol{x}} \lambda' \,|\, \lambda \in L_{\boldsymbol{x}},\ \lambda' \in \mathcal{L} \setminus L_{\boldsymbol{x}_i}\}$, and can hence be considered as *incomplete* information about an underlying calibrated label ranking. More specifically, $L_{\boldsymbol{x}}$ is consistent with the set of label rankings $E(L_{\boldsymbol{x}})$ given by those permutations $\pi \in \Omega$ that rank all labels in $L_{\boldsymbol{x}}$ higher and all labels in $\mathcal{L} \setminus L_{\boldsymbol{x}_i}$ lower than the neutral label λ_0. In the following, when we speak about a ranking, we always mean a calibrated ranking (i.e., Ω contains the neutral label λ_0).

3 Instance-Based Multilabel Classification

So far, no assumptions about the conditional probability measure $\mathbf{P}(\cdot \,|\, \boldsymbol{x})$ on Ω were made, despite its existence. To become more concrete, we resort to a popular and commonly used distance-based probability model introduced by Mallows [5]. The standard Mallows model is a two-parameter model that belongs to the exponential family:

$$\mathbf{P}(\sigma \,|\, \theta, \pi) = \frac{\exp(-\theta D(\pi, \sigma))}{\phi(\theta, \pi)} \tag{2}$$

The ranking $\pi \in \Omega$ is the location parameter (mode, center ranking) and $\theta \geq 0$ is a spread parameter.

Obviously, the Mallows model assigns the maximum probability to the center ranking π. The larger the distance $D(\sigma, \pi)$, the smaller the probability of σ becomes. The spread parameter θ determines how quickly the probability decreases, i.e., how peaked the distribution is around π. For $\theta = 0$, the uniform distribution is obtained, while for $\theta \to \infty$, the distribution converges to the one-point distribution that assigns probability 1 to π and 0 to all other rankings.

Coming back to the label ranking problem and the idea of instance-based learning, i.e., local prediction based on the nearest neighbor estimation principle, consider a query instance $\boldsymbol{x} \in \mathbb{X}$ and let $\boldsymbol{x}_1 \ldots \boldsymbol{x}_k$ denote the nearest neighbors of $\boldsymbol{x}$ (according to an underlying distance measure on $\mathbb{X}$) in the training set, where $k \in \mathbb{N}$ is a fixed integer. Each neighbor $\boldsymbol{x}_i$ is associated with a subset $L_{\boldsymbol{x}_i} \subseteq \mathcal{L}$ of labels. In analogy to the conventional settings of classification and regression, in which the nearest neighbor estimation principle has been applied for a long time, we assume that the probability distribution $\mathbf{P}(\cdot \,|\, \boldsymbol{x})$ on Ω is (at least approximately) *locally constant* around the query $\boldsymbol{x}$, so that the neighbors can be considered as a sample on the basis of which $\mathbf{P}(\cdot \,|\, \boldsymbol{x})$ can be estimated.

Thus, assuming an underlying (calibrated) label ranking, the probability to observe $L_{\boldsymbol{x}_i}$ is given by

$$\mathbf{P}(E(L_{\boldsymbol{x}_i})) \;=\; \sum_{\sigma \in E(L_{\boldsymbol{x}_i})} \mathbf{P}(\sigma \,|\, \theta, \pi) \;\;,$$

where $E(L_{\boldsymbol{x}_i})$ denotes the set of all label rankings consistent with $L_{\boldsymbol{x}_i}$. Making a simplifying assumption of independence, the probability of the complete set of observations

$\mathbf{L} = \{L_{\boldsymbol{x}_1}, L_{\boldsymbol{x}_2} \dots L_{\boldsymbol{x}_k}\}$ then becomes

$$\begin{aligned}\mathbf{P}(\mathbf{L} \,|\, \theta, \pi) &= \prod_{i=1}^{k} \mathbf{P}(E(L_{\boldsymbol{x}_i}) \,|\, \theta, \pi) \\ &= \prod_{i=1}^{k} \sum_{\sigma \in E(L_{\boldsymbol{x}_i})} \mathbf{P}(\sigma \,|\, \theta, \pi) \\ &= \frac{\prod_{i=1}^{k} \sum_{\sigma \in E(L_{\boldsymbol{x}_i})} \exp\left(-\theta D(\sigma, \pi)\right)}{\left(\prod_{j=1}^{m} \frac{1-\exp(-j\theta)}{1-\exp(-\theta)}\right)^{k}} \ .\end{aligned} \tag{3}$$

Instance-based prediction of the (calibrated) label ranking $L_{\boldsymbol{x}}$ can now be posed as a Maximum Likelihood problem, namely as finding the Maximum Likelihood estimation (MLE) of π (and θ) in (3). This problem is extremely difficult in general. Fortunately, in the context of multi-label classification, we are able to exploit the special structure of the observations. More specifically, we can show the following theorem (proof omitted).

Theorem 1: For each label $\lambda_i \in \mathcal{L}$, let $f(\lambda_i)$ denote the frequency of occurrence of this label in the neighborhood of $\boldsymbol{x}$, i.e., $f(\lambda_i) = \#\{j \,|\, \lambda_i \in L_{\boldsymbol{x}_j}\}/k$. Moreover, let $f(\lambda_0) = 1/2$ by definition. Then, a ranking $\pi \in \Omega$ is a MLE in (3) iff it guarantees that $f(\lambda_i) > f(\lambda_j)$ implies $\pi(i) < \pi(j)$.

According to this result, an optimal ranking and, hence, an optimal multi-label prediction can simply be found by sorting the labels according to their frequency of occurrence in the neighborhood. A disadvantage of this estimation is its ambiguity in the presence of ties: If two labels have the same frequency, they can be ordered in either way. Interestingly, we can remove this ambiguity by replacing the MLE by a Bayes estimation.

Theorem 2: Let $g(\lambda_i)$ denote the frequency of occurrence of the label λ_i in the complete training set. There exists a prior distribution $\mathbf{P}$ on Ω such that, for large enough k, a ranking $\pi \in \Omega$ is a maximum posterior probability (MAP) estimation iff it guarantees the following: If $f(\lambda_i) > f(\lambda_j)$ or $f(\lambda_i) = f(\lambda_j)$ and $g(\lambda_i) > g(\lambda_j)$, then $\pi(i) < \pi(j)$.

This result suggests a very simple prediction procedure: Labels are sorted according to their frequency in the neighborhood of the query, and ties are broken by resorting to global information outside the neighborhood, namely the label frequency in the complete training data (which serve as estimates of the unconditional probability of a label).

4 Related Work

Multilabel classification has received a great deal of attention in machine learning in recent years, and a number of methods has been developed, often motivated by specific types of applications such as text categorization [7–10], computer vision[11], and bioinformatics [12, 13, 10]. Besides, several well-established methods for conventional classification have been extended to the multi-label case, including support vector machines [14, 13, 11], neural networks [10], and decision trees [15].

Our interest in instance-based multilabel classification is largely motivated by the *multilabel k-nearest neighbor* (MLKNN) method that has recently been proposed in [2]. In that paper, the authors show that MLKNN performs quite well in practice. In the concrete experiments presented, MLKNN even outperformed some state-of-the-art model-based approaches to multilabel classification, including RankSVM and AdaBoost.MH [13, 16].

MLKNN is a binary relevance learner, i.e., it learns a single classifier h_i for each label $\lambda_i \in \mathcal{L}$. However, instead of using the standard k-nearest neighbor (KNN) classifier as a base learner, it implements the h_i by means of a combination of KNN and Bayesian inference: Given a query instance $\boldsymbol{x}$ with unknown multilabel classification $L \subseteq \mathcal{L}$, it finds the k nearest neighbors of $\boldsymbol{x}$ in the training data and counts the number of occurrences of λ_i among these neighbors. Considering this number, y, as information in the form of a realization of a random variable Y, the posterior probability of $\lambda_i \in L$ is given by

$$\mathbf{P}(\lambda_i \in L \,|\, Y = y) = \frac{\mathbf{P}(Y = y \,|\, \lambda_i \in L) \cdot \mathbf{P}(\lambda_i \in L)}{\mathbf{P}(Y = y)} \quad , \tag{4}$$

which leads to the decision rule

$$h_i(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \mathbf{P}(Y = y \,|\, \lambda_i \in L)\mathbf{P}(\lambda_i \in L) \geq \mathbf{P}(Y = y \,|\, \lambda_i \notin L)\mathbf{P}(\lambda_i \notin L) \\ 0 & \text{otherwise} \end{cases}$$

The prior probabilities $\mathbf{P}(\lambda_i \in L)$ and $\mathbf{P}(\lambda_i \notin L)$ as well as the conditional probabilities $\mathbf{P}(Y = y \,|\, \lambda_i \in L)$ and $\mathbf{P}(Y = y \,|\, \lambda_i \notin L)$ are estimated from the training data in terms of corresponding relative frequencies. While the estimation of the former probabilities is uncritical from a computational point of view, the estimation of the conditional probabilities can become quite expensive. Essentially, it requires the consideration of all k-neighborhoods of all training instances, and the counting of the number of occurrences of each label within these neighborhoods. Implementing nearest neighbor search in a naive way, namely by linear search, this would mean a complexity of $O(kn^2)$, where n is the size of the training data. Of course, this complexity can be reduced by using more efficient algorithms and data structures for nearest neighbor search; for example, the *all nearest neighbors* problem, i.e., the problem to find the (first) nearest neighbor for each element of a data set, can be solved in time $O(n \log(n))$ [17]. Nevertheless, the computational overhead produced by this kind of preprocessing on the training data will remain a dominating factor for the overall runtime of the method.

5 Experimental Results

This section is devoted to experimental studies that we conducted to get a concrete idea of the performance of our method. Before presenting results, we give some information about the learning algorithms and data sets included in the study, as well as the criteria used for evaluation.

Table 1. Statistics for the multilabel data sets used in the experiments. The symbol * indicates that the data set contains binary features; *cardinality* is the average number of labels per instance.

DATA SET	DOMAIN	#INSTANCES	#ATTRIBUTES	#LABELS	CARDINALITY
emotions	music	593	72	6	1.87
image	vision	2000	135	5	1.24
genbase	biology	662	1186*	27	1.25
mediamill	multimedia	5000	120	101	4.27
reuters	text	7119	243	7	1.24
scene	vision	2407	294	6	1.07
yeast	biology	2417	103	14	4.24

5.1 Learning Algorithms

For the reasons mentioned previously, our main interest is focused on MLKNN, which is arguably the state-of-the-art in instance-based multilabel ranking; we used its implementation in the MULAN package [18].[1] MLKNN is parameterized by the size of the neighborhood, for which we adopted the value $k = 10$. This value is recommended in [2], where it was found to yield the best performance. For the sake of fairness, we use the same neighborhood size for our method (Mallows). In both cases, the simple Euclidean metric (on the complete attribute space) was used as a distance function. As an additional baseline we used binary relevance learning (BR) with C4.5 (the WEKA [19] implementation J48 in its default setting) as a base learner.

5.2 Data Sets

Benchmark data for multi-label classification is not as abundant as for conventional classification, and indeed, experiments in this field are often restricted to a very few or even only a single data set. For our experimental study, we have collected a comparatively large number of seven data sets from different domains; an overview is given in Table 1.[2]

The *emotions* data was created from a selection of songs from 233 musical albums [20]. From each song, a sequence of 30 seconds after the initial 30 seconds was extracted. The resulting sound clips were stored and converted into wave files of 22050 Hz sampling rate, 16-bit per sample and mono. From each wave file, 72 features have been extracted, falling into two categories: rhythmic and timbre. Then, in the emotion labeling process, 6 main emotional clusters are retained corresponding to the Tellegen-Watson-Clark model of mood: amazed-surprised, happy-pleased, relaxing-clam, quiet-still, sad-lonely and angry-aggressive.

Image and *scene* are semantic scene classification data sets proposed, respectively, by [21] and [11], in which a picture can be categorized into one or more classes. In the scene data, for example, pictures can have the following classes: beach, sunset, foliage,

[1] `http://mlkd.csd.auth.gr/multilabel.html`

[2] All data sets are public available at `http://mlkd.csd.auth.gr/multilabel.html` and `http://lamda.nju.edu.cn/data.htm`.

field, mountain, and urban. Features of this data set correspond to spatial color moments in the LUV space. Color as well as spatial information have been shown to be fairly effective in distinguishing between certain types of outdoor scenes: bright and warm colors at the top of a picture may correspond to a sunset, while those at the bottom may correspond to a desert rock. Features of the image data set are generated by the SBN method [22] and essentially correspond to attributes in an RGB color space.

From the biological field, we have chosen the two data sets *yeast* and *genbase*. The yeast data set is about predicting the functional classes of genes in the Yeast Saccharomyces cerevisiae. Each gene is described by the concatenation of micro-array expression data and a phylogenetic profile, and is associated with a set of 14 functional classes. The data set contains 2417 genes in total, and each gene is represented by a 103-dimensional feature vector. In the *genbase* data, 27 important protein families are considered, including, for example, PDOC00064 (a class of oxydoreductases) and PDOC00154 (a class of isomerases). During the preprocessing, a training set was exported, consisting of 662 proteins that belong to one or more of these 27 classes.

From the text processing field, we have chosen a subset of the widely studied *Reuters-21578* collection [23]. The seven most frequent categories are considered. After removing documents whose label sets or main texts are empty, 8866 documents are retained where only 3.37% of them are associated with more than one class label. After randomly removing documents with only one label, a text categorization data set containing 2,000 documents is obtained. Each document is represented as a bag of instances using the standard sliding window techniques, where each instance corresponds to a text segment enclosed in one sliding window of size 50 (overlapped with 25 words). "Function words" are removed from the vocabulary and the remaining words are stemmed. Instances in the bags adopt the "bag-of-words" representation based on term frequency. Without loss of effectiveness, dimensionality reduction is performed by retaining the top 2% words with highest document frequency. Thereafter, each instance is represented as a 243-dimensional feature vector.

The *mediamill* data set is from the field of multimedia indexing and originates from the well-known TREC Video Retrieval Evaluation data (TRECVID 2005/2006) initiated by American National Institute of Standards and Technology (NIST), which contains 85 hours of international broadcast news data. The task in this data set is the automated detection of a lexicon of 101 semantic concepts in videos. Every instance of this data set has 120 numeric features including visual, textual, as well as fusion information. The trained classifier should be able to categorize an unseen instance to some of these 101 labels, e.g., face, car, male, soccer, and so on. More details about this data set can be found at [24].

5.3 Evaluation Measures

To evaluate the performance of multilabel classification methods, a number of criteria and metrics have been proposed in the literature. For a classifier h, let $h(\boldsymbol{x}) \subseteq \mathcal{L}$ denote its multilabel prediction for an instance $\boldsymbol{x}$, and let $L_{\boldsymbol{x}}$ denote the true set of relevant labels. The *Hamming loss* computes the percentage of labels whose relevance is predicted

Table 2. Experimental results in terms of Hamming loss (left) and rank loss (right).

DATA SET	MLKNN	Mallows	BR	MLKNN	Mallows	BR
emotions	0.261	0.197	0.253	0.262	0.163	0.352
genbase	0.005	0.003	0.001	0.006	0.006	0.006
image	0.193	0.192	0.243	0.214	0.208	0.398
mediamill	0.027	0.027	0.032	0.037	0.036	0.189
reuters	0.073	0.085	0.057	0.068	0.087	0.089
scene	0.087	0.094	0.131	0.077	0.088	0.300
yeast	0.194	0.197	0.249	0.168	0.165	0.360

incorrectly:

$$\textsc{HamLoss}(h) = \frac{1}{|\mathcal{L}|}\big|h(\boldsymbol{x}) \,\Delta\, L_{\boldsymbol{x}}\big|, \tag{5}$$

where Δ is the symmetric difference between two sets.

To measure the ranking performance, we used the *rank loss*, which computes the average fraction of label pairs that are not correctly ordered:

$$\textsc{RankLoss}(f) = \frac{\#\{(\lambda,\lambda') \,|\, \pi_{\boldsymbol{x}}(\lambda) \leq \pi_{\boldsymbol{x}}(\lambda'), (\lambda,\lambda') \in L_{\boldsymbol{x}} \times \overline{L_{\boldsymbol{x}}}\}}{|L_{\boldsymbol{x}}||\overline{L_{\boldsymbol{x}}}|}, \tag{6}$$

where $\pi_{\boldsymbol{x}}(\lambda)$ denotes the position assigned to label λ for instance $\boldsymbol{x}$, and $\overline{L_{\boldsymbol{x}}} = \mathcal{L} \setminus L_{\boldsymbol{x}}$ is the set of irrelevant labels.

5.4 Results

The results of a cross validation study (10-fold, 5 repeats) are summarized in Table 2. As can be seen, both instance-based approaches perform quite strongly in comparison to the baseline, which is apparently not competitive. The instance-based approaches themselves are more or less en par, with a slight though statistically non-significant advantage for our method.

As discussed in the previous section, MLKNN is expected to be less efficient from a computational point of view, and this expectation was confirmed by our experiments. Indeed, our approach scales much better than MLKNN. A typical example is shown in Fig. 1, where the runtime (total time needed to conduct a 10-fold cross validation) is plotted as a function of the size of the data; to obtain data sets of different size, we sampled from the *image* data.

6 Summary and Conclusions

According to the literature, MLKNN can be considered as the state-of-the-art in instance-based multilabel classification. In this paper, we have presented an alternative instance-based multilabel classifier, which is (at least) competitive in terms of predictive accuracy, while being computationally more efficient. In fact, our approach comes down to

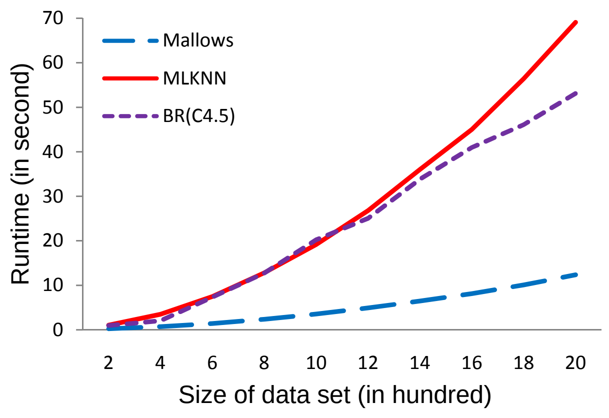


Fig. 1. Runtime of the methods on the *image* data.

a very simple prediction procedure, in which labels are sorted according to their local frequency in the neighborhood of the query, and ties are broken by global frequencies. Despite its simplicity, this approach is well justified in terms of an underlying theoretical model.

References

1. Aha, D., Kibler, D., Alber, M.: Instance-based learning algorithms. Machine Learning **6**(1) (1991) 37–66
2. Zhang, M.L., Zhou, Z.H.: ML-kNN: A lazy learning approach to multi-label learning. Pattern Recognition **40**(7) (2007) 2038–2048
3. Har-Peled, S., Roth, D., Zimak, D.: Constraint classification for multiclass classification and ranking. In Becker, S., Thrun, S., Obermayer, K., eds.: Advances in Neural Information Processing Systems. Volume 15. (2003) 785–792
4. Hüllermeier, E., Fürnkranz, J., Cheng, W., Brinker, K.: Label ranking by learning pairwise preferences. Artificial Intelligence **172**(16-17) (2008) 1897–1916
5. Mallows, C.: Non-null ranking models. In: Biometrika. Volume 44., Biometrika Trust (1957) 114–130
6. Fürnkranz, J., Hüllermeier, E., Mencia, E., Brinker, K.: Multilabel classification via calibrated label ranking. Machine Learning **73**(2) (2008) 133–153
7. Schapire, R.E., Singer, Y.: Boostexter: a boosting-based system for text categorization. Machine Learning **39**(2) (2000) 135–168
8. Ueda, N., Saito, K.: Parametric mixture models for multi-label text. In Becker, S., Thrun, S., Obermayer, K., eds.: Advances in Neural Information Processing. Volume 15., Cambridge MA, MIT Press (2003) 721–728
9. Kazawa, H., Izumitani, T., Taira, H., Maeda, E.: Maximal margin labeling for multi-topic text categorization. In Saul, L.K., Weiss, Y., Bottou, L., eds.: Advances in Neural Inf. Proc. Syst. Volume 17., Cambridge MA, MIT Press (2005)
10. Zhang, M.L., Zhou, Z.H.: Multi-label neural networks with applications to functional genomics and text categorization. In: IEEE Transactions on Knowledge and Data Engineering. Volume 18. (2006) 1338–1351

11. Boutell, M.R., Luo, J., Shen, X., Brown, C.M.: Learning multi-label scene classiffication. Pattern Recognition **37**(9) (2004) 1757–1771
12. Clare, A., King, R.D.: Knowledge discovery in multi-label phenotype data. In Raedt, L.D., Siebes, A., eds.: Lecture Ntes in Computer Science. Volume 2168., Berlin, Springer (2001) 42–53
13. Elisseeff, A., Weston, J.: A kernel method for multi-labelled classification. In Dietterich, T.G., Becker, S., Z.Ghahramani, eds.: Advances in Neural Information Processing Systems. Volume 14., Cambridge MA, MIT Press (2002) 681–687
14. Godbole, S., Sarawagi, S.: Discriminative methods for multi-labeled classiffication. In: Advances in Knowledge Discovery and Data Mining. Volume 3056 of LNCS., Springer (2004) 20–33
15. Vens, C., Struyf, J., Schietgat, L., Dzeroski, S., Blockeel, H.: Decision trees for hierarchical multi-label classification. Machine Learning **73** (2008) 185–214
16. Comite, F.D., Gilleron, R., Tommasi, M.: Learning multi-label alternating decision tree from texts and data. In Perner, P., Rosenfeld, A., eds.: Lecture Notes in Computer Science. Volume 2734., Berlin, Springer (2003) 35–49
17. Vaidya, P.: An O(n log n) algorithm for the all-nearest-neighbors problem. Discrete and Computational Geometry **4**(1) (1989) 101–115
18. Tsoumakas, G., Katakis, I.: Multi-label classiffication: an overview. International Journal of Data Warehousing and Mining **3**(3) (2007) 1–17
19. Witten, I., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. 2nd edn. Morgan Kaufmann, San Francisco, CA, USA (2005)
20. Trohidis, K., Tsoumakas, G., Kalliris, G., Vlahavas, I.: Multilabel classification of music into emotions. In: Proc. Int. Conf. Music Information Retrieval. (2008)
21. Zhou, Z.H., Zhang., M.L.: Multi-instance multi-label learning with application to scene classification. In Schlkopf, B., Platt, J., Hofmann, T., eds.: Advances in Neural Inf. Proc. Syst. Volume 19., Cambridge MA, MIT Press (2007) 1609–1616
22. Maron, O., Ratan, A.L.: Multiple-instance learning for natural scene classification. In: Proc. ICML, Madison WI (1998) 341–349
23. Sebastiani, F.: Machine learning in automated text categorization. ACM Computing Surveys **34**(1) (2002) 1–47
24. 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. ACM Multimedia, Santa Barbara, USA (2006) 421–430

Sequence Learning from Data with Multiple Labels

Mark Dredze[1], Partha Pratim Talukdar[2], and Koby Crammer[2]

[1] Human Language Technology Center of Excellence
Johns Hopkins University
Baltimore, MD 21211
mdredze@cs.jhu.edu

[2] Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
{partha,crammer}@cis.upenn.edu

Abstract. We present novel algorithms for learning structured predictors from instances with multiple labels in the presence of noise. The proposed algorithms improve performance on two standard NLP tasks when we have a small amount of training data (low quantity) and when the labels are noisy (low quality). In these settings, the methods improve performance over using a single label, in some cases exceeding performance using gold labels. Our methods could be used in a semi-supervised setting, where a limited amount of labeled data could be combined with a rule based automatic labeling of unlabeled data with multiple possible labels.

1 Introduction

Supervised learning requires large amounts of labeled training data but in many real world settings constraints imposed by cost and time for dataset construction lead to a decrease in data quality and quantity. Often times determining a single best label for an instance is difficult, especially in the case of sequence problems. One possible relaxation is to provide multiple possible training labels without choosing a single correct label. For example, multiple annotators can provide contradictory labels or automated systems can provide several good guesses for the correct label. The resulting adjudication of corpora is expensive; selecting the majority label is a good alternative, but introduces label noise.

However, instead of enforcing artificial agreement on the labels – selecting a single label a priori – all likely labels could be used by the learning algorithm. Alternatively, for some tasks, where generating a potential list of likely labels can be done in an unsupervised manner, a supervised learning algorithm could use all likely labels in learning a model.

We develop learning algorithms that are capable of handling instances with multiple possible labels along with an estimate as to the correct label. The resulting trained model tags the test data with a single correct label. Consider the task of named entity recognition where three annotators label a single sequence but two of them mislabel an organization (Figure 1). Instead of training on the majority label (incorrect in this case),

John	*studies*	*at*	*the*	*University*	*of*	*California*	*.*
PER	*O*	*O*	*O*	*ORG*	*ORG*	*ORG (0.33)*	*O*
						LOC (0.67)	

Fig. 1. A named entity training instance with multiple labels and label priors in parenthesis.

we use both labels weighted by their priors. As the model trains on the entire corpus, it can discover that the minority label is actually more probable. It then re-estimates the probabilities of the given labels and trains a new model. Over time, the algorithm shapes the data into a coherent annotation scheme from which it can learn.

In our setting of sequence learning with multiple labels, we are given a set of labels and an indication as to the probability of the given labels being correct (a prior over labels) for each training instance. Our algorithm works in an iterative fashion: first it creates a sequence model trained on all given labels weighted by their priors. Next, it updates the distribution over labels for each training example based on the likelihood assigned by the learned sequence model. This technique discovers correct labels and uses them for training. Previous work constructed an EM style algorithm for learning classification problems with multiple labels [1] and developed a Conditional Random Field model to handle missing data [2]. We extend this work and create a Multi-CRF (Section 3) that models multiple labels per instance.

However, more information i.e. access to multiple labels, need not necessarily improve learning. While our models can handle multiple labels, we ask when does such information benefit learning and when does selecting the most likely label yield superior results? We begin by analyzing these models by changing the quality and quantity of labelings in NLP data. We demonstrate that under the right conditions, our algorithm for modeling multiple labelings improves over a standard CRF.

2 Learning with Multiple Labels

In supervised learning for classification, we are provided with pairs of training examples (x) and labels (y). In the multiple label setting, we are given a set of possible labels instead of a single label for each instance. A learning algorithm for this setting is introduced in [1]. Formally, we are given i.i.d. training data $\mathcal{D} = \{x^{(i)}, S^{(i)}, \pi_y^{(i)}\}_{i=1}^{N}$, where $x^{(i)}$ is an instance, $S^{(i)}$ is a set of possible labels, and $\pi_y^{(i)}$ is a prior for each label $y \in S^{(i)}$. This allows for a separate prior for each label for every instance. The algorithm models all possible labels for each instance weighted by the probability that each label is correct. Since there is only one correct label for each instance, we want the model to favor a single label. At the same time we do not want the model to stray too far from the provided priors. The following objective function captures this intuition:

$$\ell(\theta) = \sum_{i=1}^{N} \sum_{y \in S^{(i)}} \hat{P}(y|x^{(i)}) \log \frac{\hat{P}(y|x^{(i)})}{\pi_y^{(i)}} - \sum_{i=1}^{N} \sum_{y \in S^{(i)}} \hat{P}(y|x^{(i)}) \log P(y|x^{(i)}, \theta) \quad (1)$$

where $\hat{P}(y|x^{(i)})$ is the estimated label distribution for a given instance $x^{(i)}$ and θ are the parameters of the model. Following our intuitions above, the first term corresponds to the KL divergence between the estimated label distribution and the prior over labels. This ensures that the model's estimates remain close to the given estimates.[1] The second term is the entropy of the data, corresponding to a Maximum Entropy classifier. The entropy term is modified so as to weigh each label by the estimated probability of the label being correct; the classifier is rewarded for using more likely labels. This objective is minimized iteratively using an EM algorithm: the E-step estimates label distributions, $\hat{P}(y|x^{(i)})$, by keeping model parameters fixed, while in the M-step a Maximum Entropy model learns parameters θ that maximize the entropy of the data.The E-step estimates the label distributions as

$$\hat{P}(y|x^{(i)}) = \frac{\pi_y^{(i)} P(y|x^{(i)}, \theta)}{\sum_{y' \in S^{(i)}} \pi_{y'}^{(i)} P(y'|x^{(i)}, \theta)} \tag{2}$$

for all $y \in S^{(i)}$ and 0 otherwise. Therefore, the Maximum Entropy model influences the beliefs about the correct labels subject to the given prior over labels. When $|S^{(i)}| = 1$ $\forall i$, the model reduces to a standard Maximum Entropy model.

This EM algorithm can be viewed as clustering, where each instance has a prior probability of belonging to a cluster. Instances with a single label (prior of 1) are fixed to a cluster and the algorithm clusters the remaining labels into correct and incorrect clusters. A cluster's quality depends on the ability of the CRF model to learn the associated parameters. Additionally, we can view this as self-training, a process whereby a classifier is trained iteratively on its own output. In this case, the E-step relabels the data and ensures that the algorithm's behavior is restricted since instances with a known label cannot be modified.

3 Learning CRFs with Multiple Labels

This probabilistic framework can be extended to sequence models. We are given i.i.d. training data $\mathcal{D} = \{\mathbf{x}^{(i)}, S^{(i)}, \pi_{\mathbf{y}}^{(i)}\}_{i=1}^{N}$, where $\mathbf{x}^{(i)}$ is an instance, $S^{(i)}$ is a set of labels (label sequences), and $\pi_{\mathbf{y}}^{(i)}$ is a set of priors for labels $\mathbf{y}$ for the sequence. Our goal is to learn a model that, given a sequence $\mathbf{x}$, outputs the correct label sequence $\mathbf{y}$. We use a similar objective as before (1) but extend it to sequences:

$$\ell(\theta) = \sum_{i=1}^{N} \sum_{\mathbf{y} \in S^{(i)}} \hat{P}(\mathbf{y}|\mathbf{x}^{(i)}) \log \frac{\hat{P}(\mathbf{y}|\mathbf{x}^{(i)})}{\pi_{\mathbf{y}}^{(i)}} - \sum_{i=1}^{N} \sum_{\mathbf{y} \in S^{(i)}} \hat{P}(\mathbf{y}|\mathbf{x}^{(i)}) \log P(\mathbf{y}|\mathbf{x}^{(i)}, \theta) \tag{3}$$

While classification (Section 2) used Maximum Entropy, we now use a Conditional Random Field (CRF) [3]. A CRF is defined as

$$P(\mathbf{y}|\mathbf{x}^{(i)}) = \frac{1}{Z(\mathbf{x})} \exp\Big(\sum_k \sum_t \theta_k f_k(y_t, y_{t-1}, \mathbf{x})\Big) \tag{4}$$

[1] It can be argued that the model should have the ability to deviate from the given priors without restriction, but our empirical tests found that inclusion of the first term improved model performance.

where $\{f_k(y, y', \mathbf{x})_{k=1}^K\}$ are a set of real-valued feature functions. Inserting the CRF model into (3) and applying the $\log$ yields our objective:

$$\ell(\theta) = \sum_{i=1}^{N} \sum_{\mathbf{y} \in S^{(i)}} \hat{P}(\mathbf{y}|\mathbf{x}^{(i)}) \log \frac{\hat{P}(\mathbf{y}|\mathbf{x}^{(i)})}{\pi_{\mathbf{y}}^{(i)}} - \sum_{i=1}^{N} \sum_{\mathbf{y} \in S^{(i)}} \hat{P}(\mathbf{y}|\mathbf{x}^{(i)}) \sum_t \sum_k \theta_k f_k(y_t, y_{t-1}, \mathbf{x}^{(i)}) - \sum_{i=1}^{N} \log Z(\mathbf{x}^{(i)}) \quad (5)$$

As is typical with CRFs, we add a Gaussian prior to the objective for regularization (not shown). We call the resulting model **Multi-CRF**. The second and third terms in Equation 5 are identical to a standard CRF likelihood except that it contains a weighted sum over all allowed labels of the sequence.[2] Like before, we minimize the objective with an EM algorithm. The E-step is a straight forward modification of the E-step in Section 2, yielding:

$$\hat{P}(\mathbf{y}|\mathbf{x}^{(\mathbf{i})}) = \frac{\pi_{\mathbf{y}}^{(i)} \exp(\sum_k \theta_k f_k(\mathbf{y}, \mathbf{x}^{(i)}))}{\sum_{\mathbf{y}' \in S^{(i)}} \pi_{\mathbf{y}}'^{(i)} \exp(\sum_k \theta_k f_k(\mathbf{y}', \mathbf{x}^{(i)}))}\ , \forall y \in S^{(i)},\ 0 \text{ otherwise.} \quad (6)$$

Since the normalization constant cancels out of the numerator and denominator the resulting value is easy to compute using the trained CRF. The M-step minimizes the CRF objective using standard optimization techniques (L-BFGS) [4]. The partial derivatives of the model (5) are given by:

$$\frac{\partial \ell}{\partial \theta_k} = \sum_{i=1}^{N} \sum_{\mathbf{y} \in S^{(i)}} \hat{P}(\mathbf{y}|\mathbf{x}^{(i)}) \sum_t f_k(y_t, y_{t-1}, \mathbf{x}^{(i)}) - \sum_{i=1}^{N} \sum_t \sum_{y,y'} f_k(y, y', \mathbf{x}^{(i)}) P(y, y'|\mathbf{x}^{(i)}) \quad (7)$$

The variables y, y' range over all states of the model. These derivatives are identical to the standard CRF except there is a weighted summation over allowed label sequences.

4 Evaluation

To create an environment for evaluating effects of data properties on learning, we constructed data with multiple labels similar to Jin and Ghahramani [1] who add labels predicted by a naïve Bayes classifier to training instances to create a training set with multiple labels. We produce a similar dataset for sequence learning using a Hidden Markov model (HMM). The alternate labels contain systematic errors, ie. a consistent

[2] We note that $|S^{(i)}|$ (the number of labels) can grow exponentially with the length of the sequence.

labeling, and not random errors. If we randomly permuted labels they would be easy to correct, as noted by the results of Jin and Ghahramani.

Using this approach we created datasets of varying sizes (n) and noise levels (α), where α is defined as the probability that an incorrect label will be given a greater prior than the correct label. n corresponds to the quantity of the data and α to its quality. First, we train an HMM on all available training data and label each training instance with the HMM's prediction, retaining sentences for which the prediction differs from the correct label. Next, we select the first n sentences from the training data to create datasets of varying size. We then assign a prior to each label ($\pi_{\mathbf{y}}^{(i)}$), where the correct label receives a higher prior $(1-\alpha)$ fraction of the time. We set the prior of the more likely label (max) to be $1-\alpha$ and the less likely label (min) to α. This ensures that the likelihood that the max label is correct matches the noise level of the data.

We selected two common benchmark sequence labeling tasks in the NLP community for evaluating our algorithms: the CoNLL 2003 English named entity dataset [5] and the CLASSIFIEDS segmentation data [6]. For the CoNLL dataset, we used annotations for people, locations, and organizations and created datasets of n = 200, 1k, and 5k using the given 14,042 training sentences with noise levels (α) of 0.1, 0.3, and 0.45. Results were validated on the 3,251 development instances and tested on the 3,454 test instances. [3] For CLASSIFIEDS, where only 103 training instances are provided, we created datasets of n = 10, 20, and 50 instances with noise levels (α) of 0.2 and 0.4. Results were validated on the 101 development instances and tested on the 101 test instances. We use standard orthographic features for these types of tasks [7]. For the CoNLL dataset, 5000 training instances resulted in 160k features,while in the CLASSIFIEDS dataset 50 training instances resulted in 25k features.

4.1 Results

We evaluated our CRF learning algorithm against two baselines: a CRF trained on the correct labels (*GOLD*) and a CRF trained on the most likely (maximum) label (MAX). The former indicates performance knowing the correct annotation and the latter is a heuristic used to select from multiple labels, ie. take the best label only. We evaluated several CRF multiple label algorithms on both tasks. First, we ran the Multi-CRF once without the EM algorithm, meaning that it does not reestimate label distributions ($Multi$). Our second test ran the full EM algorithm with the Multi-CRF ($Multi_{EM}$). We also included a version of the EM algorithm using MAX in the M-step (MAX_{EM}). This can improve over the baseline since it re-estimates the label distributions and relearns. We include this for comparison with $Multi_{EM}$.

For each experiment, we trained the CRF for 20 iterations, ran 7 EM iterations for MAX_{EM}, and for $Multi_{EM}$, 50 EM iterations on CoNLL and 30 EM iterations on CLASSIFIEDS. MAX_{EM} converges faster since it does not reestimate label distributions. We observed that the number of iterations mentioned above were enough for convergence. We selected the highest performing model on development data. Results

[3] Setting α as 0.5 would randomize the data; we instead use 0.45 since we assume that the data contains some indication as to the correct annotation scheme.

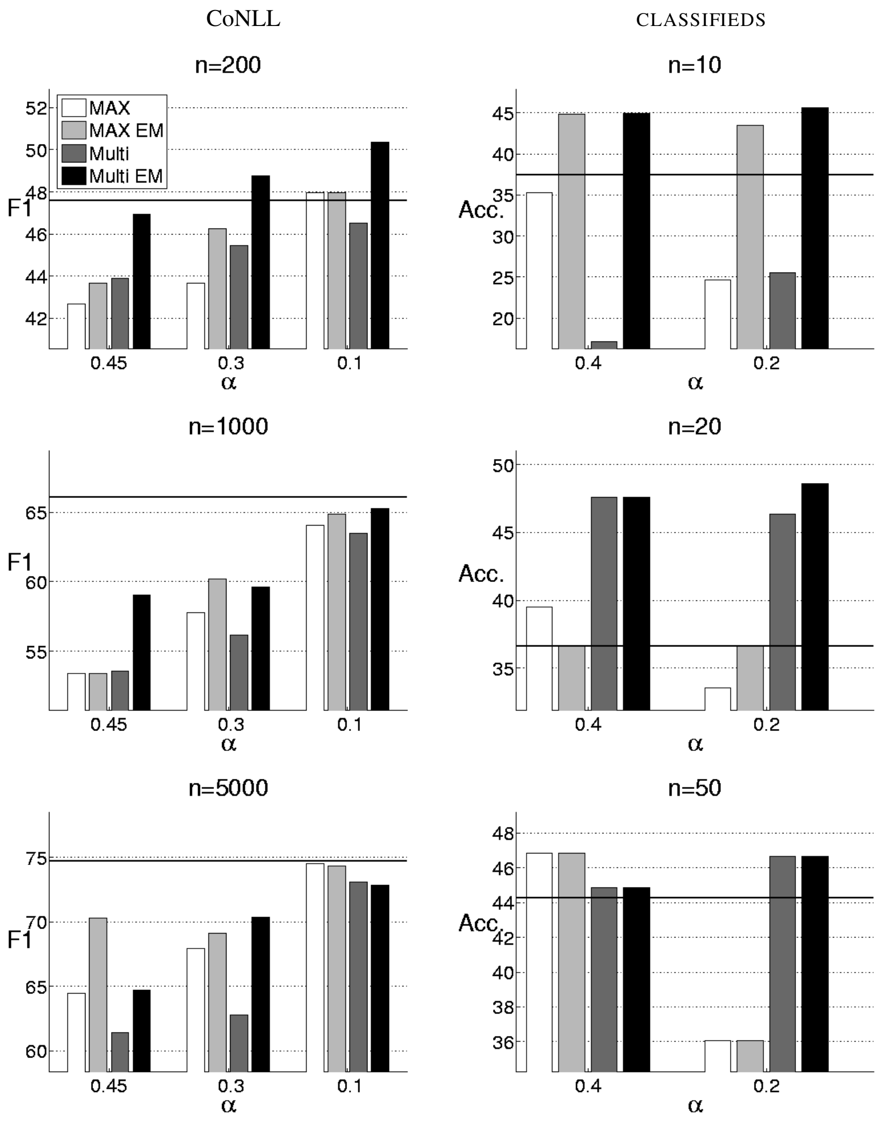


Fig. 2. Performance of the CRF learning methods on CoNLL (left)and CLASSIFIEDS (right) for increasing data set sizes (n) (top to bottom) and noise levels (α) (left to right in each figure). *GOLD* performance is indicated by a horizontal line.

for each dataset size (n) and each noise level (α) are shown in Figure 4.1 for CoNLL and CLASSIFIEDS.

Learning using multiple labels ($Multi_{EM}$) improved over taking the maximum label (MAX) most of the time, even improving over *GOLD* in some cases. While this last

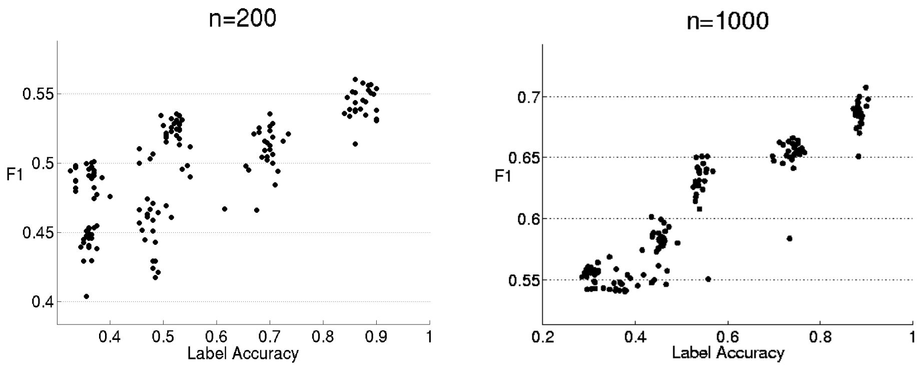


Fig. 3. The x-axis shows the agreement after each iteration between the estimated most likely label and the true gold label in the training data. The y-axis is the resulting F1 score on development data. Data points were taken after each iteration (150 points) for runs of the Multi-CRF on CoNLL $n = 200$ (left) and $n = $1k (right). Identical label agreements do not necessarily indicate identical clusterings of labels, so they can produce models with different F1 scores.

point seems strange, it could be that allowing the algorithm to influence the data can modify the data so that it is easier to learn, removing difficult examples and improving generalization performance on test data. In this way, our EM algorithm can be considered a clustering algorithm for the labels. In the case of two labels per instance, the algorithm clusters labels into the majority and minority label groups. Figure 3 shows the impact of these clusters on learning. When the labels assigned higher probability by the model are correct (improved cluster accuracy) model performance improves. Maximizing label accuracy tends to maximize F1.

5 When is Learning Successful?

While results show that multi-label learning can improve over learning with a single label, it is helpful to consider when we can expect to see such improvements. Our results indicate that two parameters effect learning: the quantity (n) and quality (α) of the training data. On one end of the spectrum, with small amounts of training data and lots of noise, low quality and quantity, our algorithms give the largest improvements. This makes sense since there is both the greatest potential for improvement (difference between MAX and *GOLD*) and significant information can be gained from the additional labels. This is exactly when one would use our methods. On the other end of the spectrum, with low noise and lots of data, both high quality and quantity, improvements are minimal and our algorithms can even hurt performance. In these cases, there is sufficient training data of high quality that multiple labels are unhelpful.

Between these two extremes, as either quality or quantity improve the baseline (MAX) approaches the performance of gold data. When a model has access to a large number of examples, it can more easily find outliers (noise) by examining many similar instances, allowing good performance with lots of low quality data. The one case

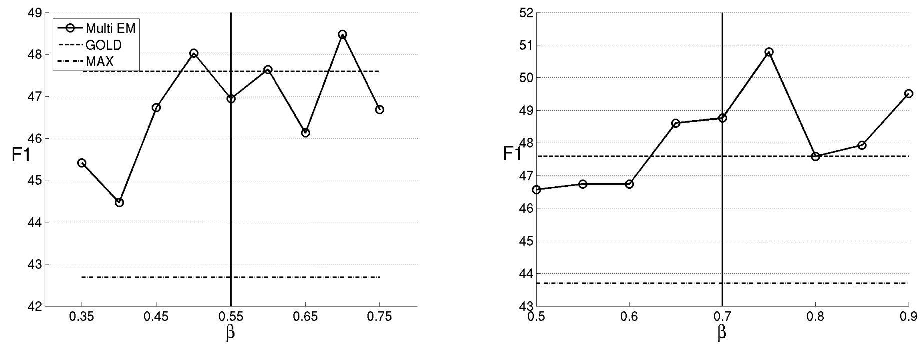


Fig. 4. The performance of a CRF $Multi_{EM}$ on CoNLL with $n = 200$, with α of 0.55 (left) and 0.7 (right). The x-axis contains varying β plotted against the resulting F1 scores. *GOLD* and MAX are shown as horizontal lines and the vertical line indicates the fixed α value.

where MAX outperforms all CRF multiple label methods is where we have the highest quantity ($n = 5000$) and quality ($\alpha = 0.1$) of data.

Noise and reliability of data has been studied in several settings [8]. In a standard theoretical result, Aslam and Decatur give a lower bound on the number of examples needed for learning using a noise level parameter [9]. As noise increases, the number of examples needed increases as well, meaning that an increased training set will offset learning errors from noise.[4] Our empirical observations are consistent with these theoretical results.

Another important point is that the efficacy of learning with multiple labels depends on the accuracy of the noise level belief. Consider the case where the algorithm is provided a majority and minority set of labels for each instance, but is told incorrectly that the majority labels are always correct ($\alpha = 0$). Obviously, the algorithm will ignore the minority labels and not improve. Conversely, if all majority labels are correct and the algorithm is told that $\alpha = .5$, it will incorrectly model the minority labels. Therefore, the algorithm should only consider alternate labels if the majority labels are incorrect, ie. the importance the algorithm assigns to minority labels should be proportional to the likelihood that they are correct.

In the experiments in Section 4.1, we assumed knowledge of the noise level and set the priors of the labels appropriately. We now test the impact of the priors by varying the max label's prior β from the true noise level α. We evaluated a CRF $Multi_{EM}$ varying β from α by 0.20 in increments of .05. Our results (Figure 4) show that different values of β impact performance but in all cases the model still outperforms the MAX baseline. It appears to be safer to be conservative, underestimating the noise level of the data. As more minority labels are ignored, the model reverts to MAX.[5]

[4] This result assumes that the noise in the data is random. While our alternative labels are not random (they are generated by an HMM), we randomly decide if it should have a higher prior than the correct label.

[5] Our observations here about the importance of β apply to the results of Jin and Ghahramani as well. While they do not consider these values, their data uses a β of 2/3 even though α is set to

Our experiments have assumed that all majority labels shared a single prior but our algorithmic formulation allows for a per-instance prior estimate. We tested a CRF $Multi_{EM}$ on a per-instance prior dataset. We selected the CoNLL $n = 200$ dataset and randomly generated an α for each instance between 0.5 and 1. We selected the correct label to be the majority label with a probability of the α for that instance. The prior of each majority label (β) was set to be α plus some random noise (up to 0.1). The resulting dataset had a different prior β for each instance which is a noisy estimate of the true noise level α for each instance. CRF $Multi_{EM}$ improved by approximately 1% over MAX on this data, showing that our algorithms can be competitive even in a per-instance prior setting.

6 Related Work

Most previous work on multi-label classification has focussed on the setting where a single instance can have multiple valid labels [10]. In contrast, the setting considered in this paper involves instances with a single valid label, though during training the instances can have multiple labels assigned to them, at most one of which is correct. This is similar in spirit to that of [1] which deals with learning from multiple labels in the *classification* setting, while the focus of this paper is to learn *structured predictors*.

A closely related area of research studies priors over parameters (instead of labels) [11]. For example, an algorithm for transfer learning by specifying priors over parameters is presented in [12]. Similarly, Raina et al. [11] compute priors over model parameters from multiple users in a transfer setting. These methods rely on the specification of priors over the model parameters, a difficult task for a human annotator. In contrast, we allow for priors over labels, which are easily specified and contain rich information about relevant features.

Recent work on *expectation regularization* (XR) [13] for classification uses a prior over labels in a corpus. This knowledge is combined with unlabeled data for effective semi-supervised learning. While the idea of providing a prior over labels is similar to our setting, there are several important differences. First, XR incorporates priors over single labels (or features), without a clear way of extending the method to interactions between multiple features. In contrast, a prior over a label in our setting can capture multiple feature interactions. Additionally, XR uses a prior over a label *type* at the corpus level, e.g. the user specifies that a particular label, *B-PER*, is likely to occur around 10% of the time in the whole corpus. On the contrary, our method defines a per-instance (and also per-position) prior over labels, which adds further granularity to the description of the data. Finally, since XR adds an additional regularization term to the objective function, we could add a similar term to our Multi-CRF for XR semi-supervised learning with multiple labels in training data. More recently, *Generalized Expectation (GE)* [14, 15] has been proposed using which one can learn from labeled features instead of labeled instances. Such expectation constraints are very useful when one can reliably label features. As in XR, GE expectation constraints are specified across all instances while the multi-label setting considered in this paper is instance specific.

0.7. While their methods appear to do well, they do not include a max label baseline. Without duplicating their experiments, we cannot determine the impact of β on their setting.

7 Conclusion

In this paper we have presented novel learning algorithms for learning structured predictors from multiple labels in the presence of noise. CRF based models improve performance on two standard NLP tasks when we have smaller amount of training data (low quantity) and when the majority labels are noisy (low quality). In these settings, the methods improve performance over using a single max label, in some cases exceeding performance using gold labels. An analysis of these results shows where multi-label learning can be most effective: when data suffers from either low quality or low quantity.

References

1. Jin, R., Ghahramani, Z.: Learning with multiple labels. In: Neural Information Processing Systems (NIPS). (2002)
2. Bellare, K., McCallum, A.: Learning extractors from unlabeled text using relevant databases. In: Workshop on Information Integration on the Web at AAAI. (2007)
3. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: International Conference on Machine Learning (ICML). (2001)
4. Nocedal, J., Wright, S.: Numerical optimization. Springer (1999)
5. Tjong, E.F., Sang, K., Meulder, F.D.: Introduction to the conll-2003 shared task: Language-independent named entity recognition. In: Conference on Natural Language Learning (CoNLL). (2003) http://www.cnts.ua.ac.be/conll2003/ner/.
6. Grenager, T., Klein, D., Manning, C.: Unsupervised learning of field segmentation models for information extraction. In: Association for Computational Linguistics (ACL). (2005)
7. McDonald, R., Pereira, F.: Identifying gene and protein mentions in text using conditional random fields. BMC Bioinformatics (S6) (2005)
8. Crammer, K., Wortman, J., Kearns, M.: Learning from data of variable quality. In: Neural Information Processing Systems (NIPS). (2005)
9. Aslam, J.A., Decatur, S.E.: On the sample complexity of noise-tolerant learning. Information Processing Letters (1996) 189–195
10. Tsoumakas, G., Katakis, I.: Multi label classification: An overview. International Journal of Data Warehousing and Mining (2007)
11. Raina, R., Ng, A.Y., Koller, D.: Constructing informative priors using transfer learning. In: International Conference on Machine Learning (ICML). (2006)
12. Marx, Z., Rosenstein, M.T., Dietterich, T.G., Kaelbling, L.P.: Two algorithms for transfer learning. In: Workshop on Inductive Transfer: 10 years later at NIPS. (2005)
13. Mann, G.S., McCallum, A.: Simple, robust, scalable semi-supervised learning via expectation regularization. In: International Conference on Machine Learning (ICML). (2007)
14. McCallum, A., Mann, G., Druck, G.: Generalized expectation criteria. Computer science technical note, University of Massachusetts, Amherst, MA (2007)
15. Mann, G., McCallum, A.: Generalized Expectation Criteria for Semi-Supervised Learning of Conditional Random Fields. In: Association for Computational Linguistics (ACL). (2008)

An Ensemble Method for Multi-label Classification using a Transportation Model

Ehud Itach, Lena Tenenboim, and Lior Rokach

Department of Information Systems Engineering, Ben-Gurion University of the Negev
Deutsche Telekom Laboratories at Ben-Gurion University of the Negev,
P.O.B. 653, Beer-Sheva 84105, Israel
{itach, lenatm, liorrk}@bgu.ac.il

Abstract. This paper proposes an ensemble method for multi-label classification. Each member of the ensemble is first associated with a small, fixed and predefined subset of labels and then a single label classifier that considers each element in the powerset of the subset, is constructed. The proposed algorithm chooses the minimum required subsets of labels at size k that covers all the possible subsets of labels at size r ($r < k$). The algorithm uses the transportation model in order to make the selection. In fact, the all covering subsets are being prepared in advance. The experimental results indicate an approximately high and stable predictive performance compared to the random subset selection approach.

1 Introduction

Researchers from various fields are facing great challenges in mining knowledge from available data due to sheer mass of information that is being generated. Supervised learning methods can be used to discover relationships between the input attributes (independent variables) and the target attribute (dependent variable). The relationship discovered is represented in a structure referred to as a model. Usually models can be used for predicting the value of the target attribute knowing the values of the input attributes.

In a single label classification task, each classifier is a function that maps an instance (data item) into one label taken from a set of disjoint labels for the purpose of prediction. Ensuring that the classes are mutually exclusive is necessary in order to guarantee that each training instance will not be associated with more than one label.

In multi-label classification tasks, the classes are not mutually exclusive. That is to say, each training instance can be associated with any subset of labels taken from the initial group of labels. Multi-label classification is needed for a wider range of applications such as text categorization [5, 10, 19] (e.g. books associated with multiple genres) and medical diagnosis [1] (e.g. patients with multiple diseases) etc.

Tsoumakas and Katakis [16] categorized multi-label classification methods into two main groups: 1. problem transformation methods. 2. algorithm adaptation methods. The first group of methods transforms the multi-label classification problem into one or more single-label classification problems. The second group adjusts known single-label classifiers into multi-label data. The main criticism about the second group of methods is that it is mostly fitted to a specific classifier (e.g. SVM, decision tree), and thus it lacks generalization. The first group, on the other hand, is suited in many cases to various classifiers. In this paper we focus on the first group of classification methods.

Two known multi-label classification methods [3, 4, 16] that are relevant to the first group of methods are: Binary Relevance (BR) [4] and Label Powerset (LP) [3].

The BR method builds independent binary classifiers for each label (λ) associated with a set of disjoint labels (L). Each classifier maps the original dataset into a single binary label with binary values $\lambda, \neg\lambda$. A given instance is associated with the label if the value is λ and not associated if the value is $\neg\lambda$. The LP method builds one classifier that refers to each unique subset as single label.

The ensemble method is a way to gather several classifiers, each different in its weakness from the other, and to integrate them into a single, strong composite model for achieving better predictive performance [12, 9]. Tsoumakas and Vlahavas [17] presented a successful ensemble method (RAKEL) for solving multi-label classification tasks. In RAKEL, each ensemble member constructs a LP classifier based on small random subset of labels. The authors showed that RAKEL achieved high predictive performance compared to BR and LP methods. The authors infer that the random selection of subsets in RAKEL may negatively affect the ensemble's performance.

The goal of this paper is to examine if the performance of RAKEL can be improved by constructing an ensemble of LP classifiers based on meaningful subsets instead of using random subsets. The meaningful subsets in our terms are subsets that promise coverage of all possible labels i.e., the subsets are specified in such a way that interactions among labels are identified. The subsets are selected in advance by solving a transportation model problem [7]. The solution of the transportation problem results in a compact ensemble that covers all possible pairs of labels. In the ensemble's classification phase, each ensemble member returns a zero-one decision about each label. An average decision of each label classification is then compared with a threshold value for the final decision about each label classification [2].

This paper presents an experiment study of two datasets. The predictive results of the suggested ensemble method are compared with the RAKEL method using the same input parameters.

The remainder of this paper is organized as follows: Section 2 describes the proposed novel ensemble algorithm. In Section 3 we discuss about the equivalence of a multi-label classification task to transportation model. In Section 4 our experimental study is described. The results are presented and discussed in Section 5. Finally, we conclude and point to future research in Section 6.

2 The Ensemble Algorithm

One of the challenges in constructing an efficient ensemble for multi-label classification is in determining the label subsets for each ensemble member. The proposed algorithm allows us to construct a compact ensemble based on a list of small, but meaningful, subset of labels.

This section starts with the matrix representation of the multi-label problem, followed by a novel method for selecting the label subsets and concludes by presenting the complete ensemble method.

2.1 Multi-label problem representation

The error-correcting output coding (ECOC) [11, 15] for solving multiclass problems inspired our method for representing a multi-label problem. As in ECOC, we create a binary matrix which includes one column for each class and each row represents a classifier that must be built. However, since ECOC is used to solve multiclass problems the matrix is also used differently.

In ECOC for each row in the matrix, a binary classifier is built such that all classes with value ’0’ in any particular row belongs to the negative class and all classes with value ’1’ belong to the positive class. With multi-label problems, on the other hand, which is the focus of this paper, the created classifier or each row refers only to the classes with value ’1’. The methods for creating ECOC matrices include rows with many ’1’s and it is impractical to enumerate all classes in that row. Instead of building a binary classifier as in ECOC, we build a multiclass classifier which enumerates all possible combinations of the selected classes. The new method we developed creates the matrix such that the amount of ’1’ in a certain row will not exceed a certain moderate value.

To illustrate the unsuitable representation of the ECOC matrix for multi-label problems, we present an example of one form of ECOC matrix representation known as *Orthogonal array* [6]. An orthogonal array $OA(n, k, d, t)$ is a matrix of k rows and n columns, with every element being one of the d values. The array has strength t if, in every t by n submatrix, the d^t possible distinct rows all appear the same number of times.

Table 1 presents an example of an orthogonal array with $d = 2$ and $t = 2$. Notice that any possible combination of any two columns of the matrix appears the exact same number of times. On the other hand, the fact that all of the labels on the first row are selected may lead to a high cost in performance when enumerating all possible labels combinations. It should also be mentioned that constructing an orthogonal array is not a simple task and in some cases even impossible.

2.2 Matrix Design for Multi-Label Classification Tasks

In this section, we suggest a new matrix design method that, on the one hand, considers correlation among labels (as in orthogonal arrays) and on the other, limits the number of selected labels in each row to a certain moderate value.

Let D be the classification domain and L be the set of labels in D (Note that $l = |L|$).

The goal of our method is to construct a binary matrix, in the form of an orthogonal array, that meets the following criteria:

1. The allowed number of selected labels in each row corresponds to the moderate value k. We use the concept of *k-labelset* [17] to refer to a labelset at size k.

Table 1. An example of an orthogonal array with $d = 2$ and $t = 2$.

	1	2	3	4	5	6	7	8	9	10	11
1	1	1	1	1	1	1	1	1	1	1	1
2	0	1	0	1	1	1	0	0	0	1	0
3	0	0	1	0	1	1	1	0	0	0	1
4	1	0	0	1	0	1	1	1	0	0	0
5	0	1	0	0	1	0	1	1	1	0	0
6	0	0	1	0	0	1	0	1	1	1	0
7	0	0	0	1	0	0	1	0	1	1	1
8	1	0	0	0	1	0	0	1	0	1	1
9	1	1	0	0	0	1	0	0	1	0	1
10	1	1	1	0	0	0	1	0	0	1	0
11	0	1	1	1	0	0	0	1	0	0	1
12	1	0	1	1	1	0	0	0	1	0	0

2. We define the concept of *matrix coverage size* with the notation r. A matrix with a coverage size r guarantees representation of any possible combination between any r labels on the matrix i.e., if $r = 2$ then we cover all pairs of labels on the matrix. If $r = 3$ then we cover all the triplets etc. Note that a labelset such as '011010' is not represented by the matrix, if there is no row in the matrix having the three selected labels. The main constraint is that a matrix should include only the **minimum** required k-labelsets that guarantee a matrix coverage at size r.

In order to represent the problem, we use a matrix at size $[mXn]$. The matrix rows represent all possible k-labelsets taken from l and the matrix columns represent all possible r-labelsets taken from l under the assumption of $r < k$. The matrix cell value at location $[i, j]$ has a value '1' in case labelset-i represents labelset-j in the solution or else has a value '0'. Each cell on the matrix has a predefined cost. The cost of a cell at location $[i, j]$ is '1' if labelset-i contains labelset-j or else it holds ∞.

To simplify the preceding presentation, we present here an illustrative example. Consider a problem with a four label dataset ($l = 4$) where the objective function goal is to return a list of minimum labelsets at size 3 ($k = 3$) that cover each possible couple of labels on the dataset ($r = 2$) at least once.

Table 2 presents the matrix that is relevant for the current problem. Each row or column in the matrix is represented by a zero-one string where character '1' denotes a label that has been taken into consideration.

Table 2. An example of a matrix representing a labelset selection problem.

Destination / Source	(1) "1100"	(2) "1010"	(3) "1001"	(4) "0110"	(5) "0101"	(6) "0011"	Total
(1) "1110"	1	1	∞	1	∞	∞	3
(2) "1101"	1	∞	1	∞	1	∞	3
(3) "1011"	∞	1	1	∞	∞	1	3
(4) "0111"	∞	∞	∞	1	1	1	3
Total	1	1	1	1	1	1	12 / 6

The rows in the matrix represent all the possible labelsets at size 3 ($m = \binom{4}{3} = 4$ options). The columns in the matrix represent all the possible labelsets at size 2 ($n = \binom{4}{2} = 6$ options). The gray part in the cells represents the cost. For example, the cost of a cell in location [1, 3] reaches infinity because labelset '1110' doesn't contain labelset '1001'. The row total is the total number of labelsets at size 2 that could be contained in labelset at each row (fixed value $\binom{3}{2} = 3$). The column total is the number of times the labelset column should be represented in the matrix (in our case it is 1).

The next phase is to solve the problem by filling the right cells on the matrix. The following presents a heuristic solution to the problem:

1. Scan the matrix in a top-down, left-right order. Locate a cell with a cost of 1 for which both of the totals (row and column) are greater than 0.
2. Increase the value of the cell by 1 and decrease both of the totals (row and column) by 1.

This iterative process is repeated until the total of all columns is zero i.e. all the destination labelsets are represented in the matrix. The solution is the list of labelsets having at least one cell with a value greater than 1. Table 3 presents the matrix from Table 2 after activating the algorithm described above. In the current example the result labelsets are: {'1110', '1101', '1011'} instead of {'1110', '1101', '1011', '0111'}.

Table 3. The cell values of Table 2 after activating the labelset selection algorithm.

Destination / Source	(1) "1100"		(2) "1010"		(3) "1001"		(4) "0110"		(5) "0101"		(6) "0011"		Total
(1) "1110"	1	1	1	1	∞		1	1	∞		∞		3/2/1/0
(2) "1101"	1		∞		1	1	∞		1	1	∞		3/2/1
(3) "1011"	∞		1		1		∞		∞		1	1	3/2
(4) "0111"	∞		∞		∞		1		1		1		3
Total	1/0		1/0		1/0		1/0		1/0		1/0		12 6

In the next phase, we construct an ensemble of LP classifiers based on the labelsets that have been selected. According to the example, the ensemble size is 3 and we train a LP classifier for each labelset in the resulting set. Later in Section 3, we examine the equivalence of the matrix design method to the well-known transportation model [7] for the purpose of representing and solving the labelset selection problem.

2.3 The ensemble method

Figure 1 presents the required steps for creating the ensemble. At the beginning, we choose the most appropriate design matrix considering the parameters- L, k, and r. For each row in the matrix, we train a LP classifier based on the corresponding labelset. As the iteration on the matrix rows ends, an ensemble of LP classifiers is constructed. It is worthwhile noting that our selection method, contrary to other methods, makes it possible to carry out the selection in the initial stages.

```
Input: Size of labelset– k, Set of labels- L, Coverage size- r,
       Training set- D, Minimum k-labelset matrix(L, k, r) – M
Output: An ensemble of LP classifiers $h_i$
1. R ← M
2. for i ← 1 to the number of rows in M do
3.     $Y_i$ ← a labelset selected from R;
4.     train an LP classifier hi X → P($Y_i$) on D;
5.     R ← R \ {$Y_i$};
```

Fig. 1. The ensemble generation phase.

Figure 2 presents the required steps for classifying a new instance x. The model of each member on the ensemble returns a binary decision about the relevance of each label in L. Based on the average decision result of each label, the new instance is associated with the corresponding label just in case the average result is greater than a user-specified threshold t.

```
Input: new instance– x, An ensemble of LP classifiers- $h_i$, Labelset size– k,
       Set of labels- L, Minimum k-labelset matrix(L, k, r) - M
Output: Multi-label classification vector Result
1.  for j ← 1 to |L| do
2.      $Sum_j$ ← 0;
3.      $Votes_j$ ← 0;

4.  for i ← 1 to the number of rows in M do
5.      forall labels $\lambda_j \in Y_i$ do
6.          $Sum_j$ ← $Sum_j$ + $h_i(x, \lambda_j)$;
7.          $Votes_j$ ← $Votes_j$ + 1;

8.  for j ← 1 to |L| do
9.      $Avg_j$ ← $Sum_j$ / $Votes_j$
10.     if $Avg_j$ > t then
11.         $Result_j$ ← 1;
12.     else $Result_j$ ← 0;
```

Fig. 2. The ensemble classification phase.

3 Equivalence to the Transportation Model

The transportation model [7] is a special type of network problem that is used for planning how to ship commodities from a source (factories) to a destination (warehouses) at minimum cost. A unit transportation cost for units transported from source i to destination j is assumed to be given.

In this section, we draw an analogy between the labelset selection method and the transportation model. We refer to sources as k-labelsets and destinations as r-labelsets. The number of possible units to be shipped from source i to destination j is equivalent to the number of appearances of a certain r-labelset in a certain k-labelset. The unit transportation cost from source i to destination j is constant if the corresponding k-labelset contains the corresponding r-labelset or else the cost is infinity.

Table 4 provides an example of a matrix representation for a transportation problem. The gray part in the cells represents the unit transportation cost of the route represented by the cell. While each source might supply a different number of units, in our problem, each k-labelset may contain the same number of r-labelsets. In a similar way, each destination may require a different amount of units whereas in our problem, we limit the number of appearances of a certain r-labelset in any k-labelsets to one.

Table 4. An example of a matrix that represents a balanced transportation problem.

Destination / Source	1	2	3	4	Supply
1	2	8	9	11	30
2	10	7	12	5	10
3	3	4	6	10	20
Demand	20	5	25	10	60

The above matrix represents a balanced transportation problem. i.e., the total supply equals the total demand. Five methods for solving balance transportation problem are suggested in [7]. Since the methods differ in regard to the basic starting solution, the ”quality” of the results differs. According to [7], Vogel’s approximation method and minimum cost method obtain the best basic starting solution. Since Vogel’s method requires heavy computations, we focused on the minimum cost method.

The minimum cost method allocates units to be shipped from sources to destinations in an iterative manner that includes two steps per iteration:

1. Locate the route (source i to destination j) with the cheapest unit transportation cost among all possible routes, i.e., discard routes that were ”treated” in previous iterations and routes that are not associated with supply nor demand.
2. Allocate as many units as possible to the located route. e.g. if the source of the route want to supply five units and the destination demand is six units, then we allocate five units (minimum$\{5,6\}$).

If we examine the equivalence of the minimum cost method to our selection method, we notice that the same iterative process is performed. The only difference between the two methods is in the scan order for a relevant cell. In the transportation model, the method selects an arbitrary available cell that has a minimum cost in the matrix. In our selection method, on the other hand, since all the available cells have the same cost, it is necessary to impose a scan order to ensure a minimum number of labelsets in the solution.

Table 5 presents the matrix from Table 4 after solving the transportation problem using the minimum cost method. In the example, the first cell that is filled is located at position [1, 1] with the minimum cost value on the matrix equaling 2. The next cell with minimum cost on the matrix is located at position [3, 1]. This cell is discarded because destination number 1 fulfills its unit's demand 20. In this way the algorithm continues until the sources and destinations have no units to supply or that can demand.

Table 5. The cells value of Table 4 after activating the minimum cost method.

Destination / Source	1		2		3		4		Supply
1	2	20	8	-	9	10	11	-	30/10/0
2	10	-	7	-	12	-	5	10	10/0
3	3	-	4	5	6	15	10	-	20/15/0
Demand	20/0		5/0		25/10/0		10/0		60

4 Experimental study

We conducted some evaluation experiments to compare the results of the transportation and RAKEL approaches. For the experiment, we developed a software package of c# classes. We used the software for two purposes: 1. to generate variation files containing all possible labelsets at size k out of n ($n >= k$). 2. to solve transportation problems and generate a minimal number of labelset files.

For the classification tasks, we used Weka [18] and Mulan [17] (a software package that is published in http://mlkd.csd.auth.gr/multi-label.html). The test was applied on the Scene [3] and Emotions [8] datasets. Those datasets have six distinct labels and each instance can be classified to one or more of these labels. We used the SMO classifier provided by Weka as our underlying base classifier for single-label classification in all ensemble models. The tests were performed using original train and test dataset splits.

For comparison purposes, we use micro-averaged F-measure and Hamming loss measures [14, 17, 13]. In the follow section, we describe the experiments and analyze the results.

4.1 Experimental parameters

The input parameters for the algorithm are:

1. l – Number of labels on the dataset under examination.
2. k – Source- subset labels. Our parameter k corresponds to the user-specified parameter k - size of the labelsets as defined in RAKEL algorithm.
3. m – Ensemble size is defined according to the output of the minimum cost method. The m parameter corresponds to RAKEL's m parameter. But, in contrast to RAKEL, the number of models m in our ensemble is constant for each k since it is defined by solution of the transportation problem.
4. r – Coverage power. In this paper we used a constant value 2, namely, the ensemble covers all pairs of labels.

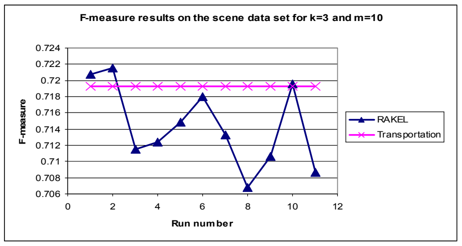
5. c – Replications. In this paper we experimented with a constant value 1, namely, each pair of labels is covered once.
6. $Threshold$ – we experimented with constant value 0.5.

We compared our results to those achieved by RAKEL with the same k and m settings. We experimented with all k values, for which the results of RAKEL and transportation methods are different. In case of datasets with 6 labels, the only meaningful values for comparison are k=3 and k=4. Note that the cases in which k=1 and k=6, correspond to building a binary model and LP model accordantly. In case of k=2 the transportation model results with all possible labelsets, so it obtains the same results as RAKEL. We did not test higher values of $k > 4$ since the number of possible models is too small.

The random nature of subset selection in the RAKEL causes different results at each execution. The results obtained by the transportation approach do not change from one implementation to another since the subsets chosen by minimal cost method are constant for a given k. Due to the potential variance in RAKEL's results between the runs, we implemented the RAKEL algorithm 11 times for each k. The transportation algorithm was executed once for each k.

5 Results and Discussion

This section presents the results of our experiments. Figures 3 and 4 present micro-averaged F-measure and H-Loss results on the scene data set for $k = 3$ and $k = 4$. The graphs show that the results of the RAKEL approach, as expected, vary between the runs due to the random selection of subsets. The results of the transportation approach are close to the highest values achieved by RAKEL. Only in few runs did RAKEL provide better results than those achieved by the transportation method. In these cases, the difference between their values is relatively very small. On the other hand, in many runs RAKEL's results are worse than those of the transportation approach. In these cases, the differences between them are higher. It should also be noticed that RAKEL's average Hamming loss result for $k = 3$ and $m = 10$ is a little bit better than the one achieved by the transportation approach. This result is marginal due to the robustness of the transportation approach.



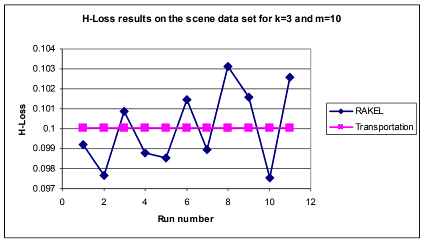


Fig. 3. F-measure and Hamming-loss results on the scene data set for k=3 and m=10.

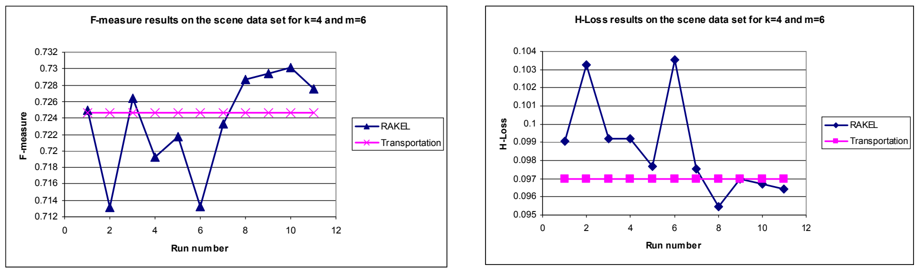


Fig. 4. F-measure and Hamming-loss results on the scene data set for k=4 and m=6.

Figures 5 and 6 present the results for the emotions data set for $k = 3$ and $k = 4$. Also, here the results of the RAKEL approach vary between runs. In the case of $k = 3$, the results of the transportation approach are better than those achieved by RAKEL in all 11 runs. The case of $k = 4$ is similar to that described in the scene data set results: the results of the transportation approach are better than those achieved by most of the RAKEL runs.

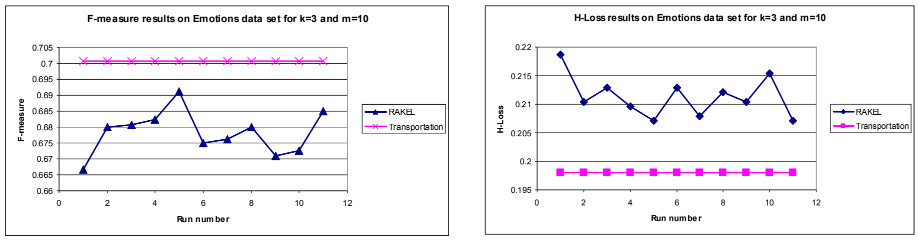


Fig. 5. F-measure and Hamming-loss results on the emotions data set for k=3 and m=10.

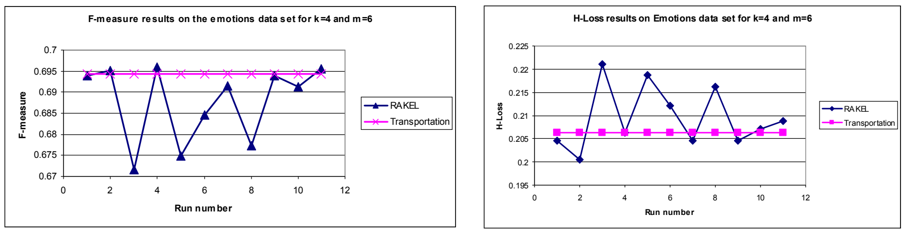


Fig. 6. F-measure and Hamming-loss results on the emotions data set for k=4 and m=6.

6 Conclusions and Future Work

In this paper we presented a new ensemble method for multi-label classification based on solution to the transportation problem. We showed that this method can determine the labelsets for each ensemble member. Specifically, the method finds a minimal number of label sets of predefined size in relation to the chosen number of labels.

Results of the experimental study show that the method performs in a highly efficient and stable manner, in many cases better than RAKEL. The proposed method resolves two main problems of the RAKEL algorithm: (1) random selection of labelsets and a large variation in the result values; and (2) the need to define the number of models in an ensemble.

In our future work, we plan to test our approach on additional datasets and include other input parameters. Moreover, we will examine more powerful coverage schemes, such as covering all triplets, and compare the results to other multi-label classification methods.

References

1. Clare A., King R.D.: Knowledge Discovery in Multi-label Phenotype Data. Lecture Notes in Computer Science, Vol. 2168, Springer, Berlin (2001).
2. Bauer, E. Kohavi, R.: "An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants". Machine Learning, 35: 1-38, 1999.
3. Boutell, M. Luo, J. Shen, X. Brown, C.: Learning multi-label scene classification. Pattern Recognition Volume 37, Issue 9, Pages 1757-1771 (2004).
4. Brinker, K., Furnkranz, J., Hullermeier, E.: A unified model for multilabel classification and ranking. Proceedings of the 17th European Conference on Artificial Intelligence (ECAI '06), Riva del Garda, Italy pages: 489-493 (2006)
5. Domonkos, T. Gyiorgy, B.: Experiments with multi-label text classifier on the reuters collection. Conference on Computational Cybernetics (ICCC03), Siofok (2003).
6. Hedayat A.S, N. J. A. Sloane N.J.A and Stufken J.: Orthogonal Arrays: Theory and Applications. Springer-Verlag, NY. (1999).
7. Imam, T. Elsharawy, G. Gomah, M. Samy I. : Solving Transportation Problem Using Object-Oriented Model: IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.2 (2009).
8. Trohidis, K., Tsoumakas, G., Kalliris, G., Vlahavas, I.: Multilabel Classification of Music into Emotions. In: Proc. 2008 International Conference on Music Information Retrieval (ISMIR 2008), pp. 325-330.
9. Maimon O., Rokach L., Decomposition Methodology for Knowledge Discovery and Data Mining. In O. Maimon and L. Rokach (Eds), Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers, pp. 981-1003. Springer, 2005.
10. McCallum, A.: Multi-label text classification with a mixture model trained by em. In: Proceedings of the AAAI' 99 Workshop on Text Learning. (1999)
11. Rayid Ghani, Using Error-Correcting Codes for Text Classification, Proceedings of the Seventeenth International Conference on Machine Learning, p.303-310, June 29-July 02, 2000
12. Rokach, L. Maimon, O.: Ensemble Methods for Classifiers. Data Mining and Knowledge Discovery Handbook. Part VI Pages 957-980, Springer US (2005).
13. Rokach L., Maimon O., Arbel R. (2006), Selective Voting-Getting More for Less in Sensor, International Journal of Pattern Recognition and Artificial Intelligence 20(3):329-350.

14. Arbel, R. and Rokach, L., Classifier evaluation under limited resources, Pattern Recognition Letters, 27(13): 1619–1631, 2006.
15. T. G. Dietterich and G. Bakiri. Solving Multiclass Learning Problems via Error-Correcting Output Codes. Journal of AI Research, 2 (1995), pp. 263-286.
16. Tsoumakas, G., Katakis, I.: Multi-label classification: An overview. International Journal of Data Warehousing and Mining3 p: 1-13 (2007)
17. Tsoumakas, G. Vlahavas, V.: Random k-Labelsets: An Ensemble Method for Multilabel Classification. The 18th European Conference on Machine Learning (ECML) (2007).
18. Frank, E., Hall, M.A., Holmes, G., Kirkby, R., Pfahringer, B., Witten, I.H.(2005). Weka: A machine learning workbench for data mining. In O. Maimon and L. Rokach (Eds), Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers, pp. 1305-1314. Springer, 2005.
19. Yang, Y.: An evaluation of statistical approaches to text categorization. Journal of Information Retrieval 1 p: 78-88 (1999)

Curve prediction with Kernel Regression

Damjan Kužnar, Martin Možina, and Ivan Bratko

University of Ljubljana, 1000 Ljubljana, Slovenia,
{damjan.kuznar|martin.mozina|ivan.bratko}@fri.uni-lj.si

Abstract. In this paper, we present a method for learning from multiple continuous target values. In contrast to most similar methods, we assume a strong correlation between the target values, as these values together form a curve. At its core, the method uses the Metric Learning for Kernel Regression (MLKR) algorithm that was extended for prediction of multiple values. We also present an additional extension in form of ridge regularization to achieve more robust optimization. Evaluation on artificial datasets is provided, together with an application on Fiat domain.

1 Introduction

Curve prediction is a specialization of multiple continuous target values prediction, where curve can be any numeric sequence obtained from a continuous function. An important implication of this is that neighboring target values exhibit strong correlation.

Since the we are dealing with multiple target values, most standard machine learning methods (e.g. regression tree), where the data is generalized into a model, are not applicable. Exception are instance based methods, where prediction is usually made by averaging the target value, that can easily be extended to predict multiple target values. One such instance based method that was used in this research is *kernel regression*. Methods like *k-nearest neighbors* are not appropriate since their loss function (error) is not continuous (changing the learning parameters slightly does not necessarily result in change of error), which is an important property that is exploited in this paper.

Recent progress in metric learning [1] inspired us to apply the approach to learning from multiple continuous target variables. Metric learning can be understood as transformation of the input space in such a way to move together similar samples and to further separate the dissimilar ones. In this paper we propose a novel algorithm for learning from multiple continuous target variables using the improved kernel regression method. Prediction is made by computing the weighted average of the train samples target values, with focus of this paper being on setting the optimal weights.

This paper is organized as follows: Section 2 defines the problem and establishes the terminology used in the paper. In Section 3 a detailed description of the proposed method is given. Section 4 reports evaluation results on artificial dataset. In Section 5 we describes the application of proposed method on Fiat domain and we conclude in Section 6.

2 Problem statement

Focus of this paper is a machine learning task with a goal of building a prediction model that is capable of predicting multiple continuous target variables with high level of correlation, more precisely, a curve. The learning problem can be defined as:

* Having a set of learning examples described with attributes $X = \{x_1, ..., x_k\}$ and class values $Y = \{y_1, ..., y_j\}$, where y_i and y_{i+1} are close enough (a curve),
* find a function f that translates from X to Y:

$$f : X \rightarrow Y$$

The desired property of f is to predict multi target values as a curve, namely the predictions of two adjacent class values need to be close enough. A possible approach assuring this property would be kernel regression, since it computes the prediction as a weighted sum of classes in learning examples. The prediction of kernel regression on examples where class is a curve will also be a curve.

Proof. Assuming we have two learning examples. Let (a_1, b_1) be two adjacent class values of the first example with a small difference $|a_1 - b_1| < \varepsilon$ and (a_2, b_2) the two adjacent values of the second example $|a_2 - b_2| < \delta$. Then, the maximal difference between the weighted sum of these values $|k_1a_1 + k_2a_2 - k_1b_1 - k_2b_2|$ is less than $(\varepsilon k_1 + \delta k_2)$, which is still small providing that ε and δ are small. The proof can be easily generalized to more than two examples.

3 Method

The proposed method is an extension of *Metric Learning for Kernel Regression* (MLKR) algorithm as introduced in [1]. We begin by giving an overview of MLKR before introducing our extensions.

3.1 MLKR

MLKR builds on standard kernel regression algorithm that computes the target value prediction $\hat{y}$ of test instance $\boldsymbol{x}$ by calculating the weighted average of target variables from training dataset [2].

$$\hat{y} = \frac{\sum_i y_i k_i}{\sum_i k_i}, \tag{1}$$

where $k_i = k(\boldsymbol{x}, \boldsymbol{x_i}) \geq 0$ is a kernel function.

The weight of each training instance is computed using a kernel function, which typically decays rapidly with distance between the test and training instance. Consequently, the estimated test instance has the strongest dependence on the nearby training instances. Commonly, kernel regression uses Euclidean distance for distance metric and Gaussian kernel for computing the weights, that decay exponentially with squared distance. The Gaussian kernel is usually defined as:

$$k_{ij} = \frac{1}{\sigma\sqrt{2\pi}} e^{\left(\frac{-d(\boldsymbol{x_i}, \boldsymbol{x_j})}{\sigma^2}\right)}, \tag{2}$$

where d is squared distance between two examples and is defined as:

$$d(\boldsymbol{x_i}, \boldsymbol{x_j}) = \textit{diff}(\boldsymbol{x_i}, \boldsymbol{x_j})^T \textit{diff}(\boldsymbol{x}_i, \boldsymbol{x_j}), \tag{3}$$

where $\textit{diff}_i$ is a custom attribute difference function.

There is an important drawback of the standard kernel regression method due to the use of *a priori* distance metric (e.g. Euclidean distance metric) on the input space, which may not be particularly relevant for the regression task at hand. For example, if certain input features are completely irrelevant to the regression task (e.g. random noise attributes), they ideally should not contribute at all to the distance metric, whereas a Euclidean distance would ascribe to them as much weight as the most significant features.

MLKR addresses this problem by using the Mahalanobis distance metric [3], that is a generalization of Euclidean distance, and the squared distance between two vectors x_i and x_j is defined as:

$$d(\boldsymbol{x_i}, \boldsymbol{x_j}) = \textit{diff}(\boldsymbol{x_i}, \boldsymbol{x_j})^T \mathbf{M} \textit{diff}(\boldsymbol{x}_i, \boldsymbol{x_j}), \tag{4}$$

where $\mathbf{M}$ can be any symmetric positive semidefinite real matrix. Using identity matrix as $\mathbf{M}$ gives the standard Euclidean metric. However, determining the optimal $\mathbf{M}$ is a non-trivial task. Weinberg proposes a novel approach for determining the optimal $\mathbf{M}$ matrix by directly minimizing the loss function

$$L = \sum_i (\hat{y}_i - y_i)^2. \tag{5}$$

First, the positive semidefinite constraint of matrix $\mathbf{M}$ needs to be removed, since it poses optimization problems. This is done by performing a matrix decomposition

$$\mathbf{M} = \mathbf{A}^T \mathbf{A}, \tag{6}$$

where A is an unconstrained matrix. Mahalanobis metric (4) can now be rewritten as

$$d(\boldsymbol{x_i}, \boldsymbol{x_j}) = \parallel \mathbf{A} \textit{diff}(\boldsymbol{x}_i, \boldsymbol{x}_j) \parallel \tag{7}$$

which allows us to calculate gradient using the equation

$$\frac{\partial L}{\partial \mathbf{A}} = 4\mathbf{A} \sum_i (\hat{y}_i - y_i)) \frac{\sum_{j \neq i} (\hat{y}_i - y_j) k_{ij} \boldsymbol{x}_{ij} \boldsymbol{x}_{ij}^T}{\sum_{j \neq i} k_{ij}}, \tag{8}$$

where $\boldsymbol{x}_{ij} = \textit{diff}(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

Gradient can now be used in any gradient based method such as gradient descent, conjugate gradient, stochastic gradient or BFGS. For this paper, gradient descent method was chosen.

3.2 Curve prediction with MLKR

We now present the extension of MLKR for curve prediction (C-MLKR). As stated before, we wish to predict the entire curve instead of a single value. Therefore, target value is now a vector $\boldsymbol{y}_i \in \mathbb{R}^m$. To accommodate this change, we need to rewrite the loss function (5) as

$$L_2 = \sum_i (\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i)^T (\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i). \tag{9}$$

This results in a new equation for gradient calculation

$$\frac{\partial L_2}{\partial \mathbf{A}} = 4\mathbf{A} \sum_i (\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i)) \left(\begin{bmatrix} \sum_{j \neq i} (\hat{\boldsymbol{y}}_{i_1} - \boldsymbol{y}_{j_1}) \mathbf{W}_{ij} \\ \vdots \\ \sum_{j \neq i} (\hat{\boldsymbol{y}}_{i_m} - \boldsymbol{y}_{j_m}) \mathbf{W}_{ij} \end{bmatrix} \frac{1}{\sum_{j \neq i} k_{ij}} \right), \tag{10}$$

where $\mathbf{W}_{ij} = k_{ij} \boldsymbol{x}_{ij} \boldsymbol{x}_{ij}^T$.

This two modifications now allow us to predict the entire curve.

3.3 Ridge C-MLKR

During the conducted empirical tests, we noticed that in some domains gradient optimization leads to very large absolute values of matrix $\mathbf{A}$. After multiplying the difference $diff(x_i, x_j)$ with such an $\mathbf{A}$, we get large distances which, after applying the Gaussian kernel, result in weights equal or very close to zero. The method selects such high values of A, since they fit the learning data well. However, a great increase of parameters in A can sometimes bring a relatively small improvement in terms of accuracy on learn data, what suggests a possible over-fitting.

To overcome this problem, we propose an extension in form of ridge regularization approach [5]. In general, ridge regression is a method for penalizing the complexity of the model by introducing penalty for setting the parameters of the model too high. In our setting, we penalize such solutions of matrix $\mathbf{A}$, that have large absolute values. Taking this into consideration, we now rewrite the loss (9) function as

$$L_3 = \sum_i (\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i)^T (\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i) + \alpha (a_{11}^2 + a_{12}^2 + \ldots + a_{kk}^2), \tag{11}$$

where a_{ij} is a value from i-th row and j-th column of matrix $\mathbf{A}$ and α is a user tunable ridge regression coefficient, where greater α results in bigger complexity penalties. New loss function requires a slight modification of gradient equation

$$\frac{\partial L_3}{\partial \mathbf{A}} = \frac{\partial L_2}{\partial \mathbf{A}} + 2\mathbf{A}\alpha. \tag{12}$$

The ridge parameter α is usually automatically optimized with the internal cross-validation technique. However, due to time-consuming optimization process, this would take too long to make it practically usable, therefore we contemplated what would be a sensible α value. The idea was to simulate statistical testing with ridge regression, where a certain change in the A matrix would require a certain improvement of the loss function. In this paper, we shall always set the α value as:

$$\alpha = 3.96 \frac{L_{euc}}{n}, \tag{13}$$

where L_{euc} is the prediction error of kernel regression with Euclidean distance and n is the number of learning examples. This value for α assures that, if a certain value in matrix A changes from 0 to 1, then the decrease of the prediction error should be significantly lower than L_{euc} with significance 0.05, if measured with the F-test.

4 Evaluation on artificial data sets

Given the complexity of the multi-label regression task and the lack of suitable publicly available datasets, we provide an evaluation on an artificial dataset. The artificial dataset contains three attributes a_1, a_2 and a_3 that are randomly drawn from interval [0, 5] with uniform distribution. The response curve (the class attribute) is then computed given the following function

$$f(t) = \sin(a_1 t) + \cos(a_2 t) a_3, \tag{14}$$

where $t\varepsilon[0, 2]$. The resulting class value is a vector $(f(t1), ..., f(tn))$, where n is the number of time series samples - greater n value corresponds to a greater sampling rate. In our case $n = 60$, since that mimics the characteristics of the Fiat wind noise domain dataset. We constructed three additional dataset by introducing three types of noise to the previous artificial dataset. First two are constructed using the following equations

$$f_{noise_1}(t) = f(t) + \varepsilon \tag{15}$$

$$f_{noise_2}(t) = \sin((a_1 + \varepsilon)t) + \cos((a_2 + \varepsilon)t)(a_3 + \varepsilon), \tag{16}$$

where ε is a random value drawn from uniform distribution on interval [-1,1]. The third dataset is the same as noise free dataset with additional 10 noise attributes drawn from interval [0, 5] with uniform distribution.

We compare the C-MLKR to the standard kernel regression (using Euclidean distance metric) and k-nearest neighbor method using the 3-fold cross validation evaluation method [4]. Reported values are computed using *root mean square error* (RMSE) which is defined as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} \mathrm{Ed}(\hat{\boldsymbol{y}}_i, \boldsymbol{y}_i)}{n}}, \tag{17}$$

where Ed is Euclidean distance.

Table 1 shows the results of evaluation.

Table 1. Evaluation results on artificial dataset

	f	f_{noise_1}	f_{noise_2}	f_{noise_3}
KNN (k=1)	14.25	14.89	48.57	81.14
KNN (k=5)	11.36	11.04	**22.92**	32.70
Kernel regression	44.14	40.86	43.41	44.14
C-MLKR	**4.76**	8.57	24.97	8.96
Ridge C-MLKR	5.12	**5.92**	23.12	**5.37**

We can see that in the noiseless domain the C-MLKR method performs best. However, with the introduction of noise (any type), the C-MLKR with ridge regularization achieves better results, which supports our intuition that basic C-MLKR can overfit to the learning examples. Interestingly, Ridge C-MLKR's error did not increase by adding random attributes, probably, as ridge regression forced the weights given to these attributes down to 0. Both kernel methods with metric learning, C-MLKR and Ridge C-MLKR, perform better than other methods. We can clearly see the improvement over standard kernel regression and nearest neighbor method. Figure 1 graphically shows the performance of different methods and gives a visual confirmation of C-MLKR improvement over standard methods.

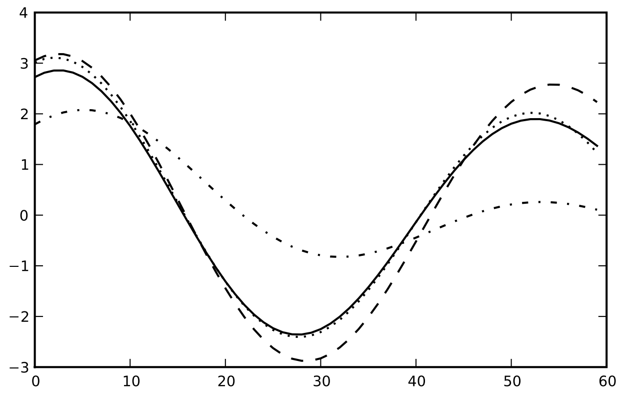


Fig. 1. Classification for datasets f_{noise_3}, where dotted line is the actual curve, solid line is the Ridge C-MLKR prediction, dashed line is the C-MLKR prediction and dash-dot line is the kNN (k=5) prediction.

5 Application on Fiat wind noise domain

Fiat non-public wind noise domain was the main motivation for development of C-MLKR algorithm. It contains data obtained from performing various aerodynamic tests on a vehicle in a wind tunnel stationed at the Fiat Research Center (Centro Ricerche Fiat). Usually a wind noise response of a single vehicle components is recorded in form of an audio frequency spectrum. By examining the spectrum, experts can then determine whether a certain component is critical for the overall reduction of the wind noise. For overall reduction of wind noise all the critical components need to be improved.

Since setting up the wind tunnel tests is time and cost expensive the task is to predict the components responses without performing the actual test. More precisely, the goal is to predict the complete audio spectrum of wind noise (vector of 6 sound pressure values averaged from 60 values) for a given vehicle component (e.g. door sealing, windshield wipers, ...). The entire vehicle setup is described with following attributes: component description, microphone description, vehicle segment, original vehicle and vehicle shape noise spectrum.

Component description is a textual description of setup, stating which component (one of approximately 30 different components) is being tested and is very important for determining which component was tested. Microphone description is also a textual description of microphone position (8 possible positions). It is important to note that these two attributes are textual. This makes them difficult to use. To overcome the problem complexity, we have chosen to build a text classifier that classifies description by the components being used in the setup. This is easily integrated into the C-MLKR framework, since the diff function allows us to specify custom difference function for every attribute.

Vehicle shape spectrum is a frequency spectrum recorded using a vehicle where all components noise is removed (e.g. by removing antenna) components do not generate noise and only noise from vehicle body shape is present. Shape vehicle can be considered as an vehicle setting, where all the components are ideal. On the other hand, the original vehicle spectrum is recorder using the original vehicle, where all components contribute to the wind noise. We handle the spectrum as a single attribute and define the custom $\textit{diff}_{spectrum} : \mathbb{R}^n \rightarrow \mathbb{R}$ function as:

$$\textit{diff}_{spectrum}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \mathrm{Ed}(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{18}$$

where Ed is Euclidean distance.

Vehicle segment is a nominal attribute classifying vehicle into different vehicle segments. For example, A is for small vehicles or LCV for light commercial vehicles.

We evaluated C-MLKR and Ridge C-MLKR using the leave-one-vehicle-out technique that is similar to leave-one-out, with difference being that we put all examples for a given vehicle in the test dataset. We compare the results to the standard kernel regression (Table 2).

Table 2. Evaluation results on Fiat domain

	Fiat domain
Kernel regression	2.92
C-MLKR	2.84
Ridge C-MLKR	2.82

As we can see, improvement over standard kernel regression are not satisfactory. This is due to complexity of the Fiat domain, where empirical study has shown that some attributes (classification of component and microphone description) contain noise. This means that target values are associated with wrong component name, thus leading to similar problems as with artificial dataset f_{noise_2}, where Ridge C-MLKR improvement is diminished. Moreover, it is necessary to note that all types of noise (f_{noise_1}, f_{noise_2} and f_{noise_3}) are present in Fiat domain, making it a very difficult domain. However, this is an ongoing research and we expect better results in the future.

6 Conclusions

We presented a novel method C-MLKR for prediction of multiple continuous target variables with strong correlation. The main novelty of this paper is the extension of MLKR (metric learning for kernel regression) to the multiple continuous target learning problem. According to our experiments on artificial data sets, the method performs better than KNN methods and kernel regression with Euclidean distance for weight estimation. Furthermore, we extended the algorithm with ridge regularization, which turned out to be useful in domains containing noise.

As future work, we are considering the option of dataset clustering to improve the speed performance. Samples can be clustered and only one representative sample is then used for testing if cluster contains similar samples. In case of dissimilarity the cluster would be ignored, since the sample weight would most probably equal zero (or very close to zero). However, in case of similarity, the entire cluster is used to compute the weighted average.

With respect to Fiat domain, we plan to improve the *diff* function between different types of components and microphones. At the moment there are approximately 30 different components, where difference is 0 if the are equal and 1 otherwise. However, by visually inspecting spectra, we noticed that some components have a similar effect on noise and their difference should therefore be less that 1.

References

1. Weinberger, K. Q.: Metric Lerning for Kernel Regression. Proceedings of the 11th AISTATS, (2006)
2. Benedetti, J. K.: On the nonparametric estimation of regression functions. Journal of the Royal Statistical Society 39, 248–253, (1977)
3. Mahalanobis, P.: Normalization of statistical variates and the use of rectangular co-ordinates in the theory of sampling distributions. Sankhay, 3, 1–40, (1937)
4. Plutowski, M.. Survey: Cross-validation in Theory and in Practice. Research Report. Dept. of Computational Science Reserach, David Sarnoff Reserach Center, Princeton, New Jersey, (1996)
5. Hoerl, A. E., Kennard, R. W.: Ridge regression: biased estimation for nonorthogonal problems. Technometrics, 12, 55–67, (1970)

Generating Synthetic Multi-label Data Streams

Jesse Read, Bernhard Pfahringer, and Geoff Holmes

Department of Computer Science
The University of Waikato
Hamilton, New Zealand
{jmr30,bernhard,geoff}@cs.waikato.ac.nz

Abstract. There are many available methods for generating synthetic data streams. Such methods have been justified by the need to study the efficacy of algorithms on a theoretically infinite stream, and also a lack of real-world data of sufficient size. Although multi-label classification has attracted considerable interest in recent years, most of this work has been carried out in the context of a batch learning environment rather than a data stream. This paper makes an in-depth analysis of multi-label data, and presents a general framework for generating synthetic multi-label data streams.

1 Introduction

A multi-label data stream is a data stream with the same properties as multi-label data. Multi-label learning problems, where an instance is assigned multiple labels from a finite set of labels, have received considerable attention in the machine learning literature, but prior work focusses almost exclusively on a batch learning environment with train-test or cross-validation scenarios. The problem of multi-label data streams has received much less attention.

Many real world practical problems involve data which can be considered as a multi-label data stream. For example news articles, e-mails, RSS-feeds, newsgroups, bookmarking, and medical text classification.

Labels can be considered as subject categories, tags, author names, or even diagnoses (in the case of the medical domain), as long as the set of labels is finite and known at the time of classification. Instances will always arrive in time order. Classification in such an environment involves an emphasis on efficiency and adaptivity:

- **Incremental learning**: examples processed one at a time; must be able to predict as new instances become available
- **Efficiency**: limited amount of time and memory; able to handle large volumes of new instances
- **Adaptivity**: must be able to handle to concept drift

Despite the ubiquitous presence of multi-label data streams in the real world, they can rarely be easily assimilated on a large scale with both labels and timestamps intact and there may issues with sensitive data – for example with e-mail, personal bookmarking, and medical text corpora. In many cases, in-depth domain knowledge may be necessary to determine and pinpoint changes to the concepts represented by the data.

Hence the reasons to generate synthetic multi-label data streams are to:

- increase the pool of multi-label stream data and thereby also increase the depth of analysis and conclusions which can be drawn in respect to the performance of various algorithms;
- allow a theoretically infinite data stream; and
- help conduct specific analysis of incremental multi-label algorithms.

This paper involves an in depth study of multi-label data, and presents a framework for generating synthetic multi-label data streams in order to facilitate the study and evaluation of multi-label algorithms in this area.

2 Prior Work

The notation to define a multi-label data stream is as follows.

- Let $\mathcal{X} = \mathbb{R}^d$ denote the input space
- Let $X \in \mathcal{X}$ be an *instance*
- Let $L = \{l_1, l_2, \cdots, l_N\}$ denote the finite label set
- Let $l_i \in L$ be a single label
- Let $S = (l_1, l_2, \cdots, l_N) \in \{0,1\}^N$ be a *label subset* representing $S \subseteq L$ where:

$$S[i] = \begin{cases} 1 & \text{if } l_i \in S \\ 0 & \text{if } l_i \notin S \end{cases}$$

- Let $d = (X, S)$ be a multi-label example consisting of an instance and relevant labels
- Let $D = d_0, d_1, \cdots$ be a theoretically infinite stream of multi-label examples

2.1 Synthetic Multi-label Data

Generating synthetic data streams has been investigated in the past for single-label data. The work in [4] provides the `MOA` framework which contains a variety of methods for the generation and classification of *single-label* data streams. This is expanded by [1] which additionally considers concept drift, as opposed to simply an incremental context. There are also numerous examples of purpose-specific multi-label data being generated.

The authors of [10] generate a multi-label synthetic dataset with three labels and two features. The examples pertaining to certain labels are associated with certain Gaussian distributions. Cai [2] uses a tree structure with random weight vectors generated for each node. Park and Fürnkranz [5] generate data using a number of labels using a set of pairwise constraints. Random permutations are generated which satisfy this set, which are in turn decomposed into binary pairwise preferences.

Kirchenko's [3] synthetic data involves a special hierarchical case where inner nodes represent labels. Synthetic data is generated by building a balanced tree hierarchy and allocating three binary attributes, with 10 training and 5 test instances generated for each label.

Overall, prior work for generating synthetic multi-label serves only to highlight certain characteristics of the algorithms that the authors present. The data usually contains as few as two or three features and labels, relatively few examples, and was never intended for large scale multi-label evaluation. More importantly, none of these data generation techniques are for creating data stream contexts, which is a main focus of this paper.

Not yet mentioned in the literature is the idea of using clustering to create multi-label data where cluster centers represent labels. This would presume the use of a clustering algorithm which can supply probabilities that an instance belongs to each cluster so that a threshold could be used to influence different degrees of multi-labelling. Any data source could be used if the original time order can be maintained. A related possibility would be to use a time-ordered single-label dataset and to reclassify using this same ranking-and-threshold method.

The advantage of these techniques is to have data with underlying real-world concepts. However, the stream cannot be theoretically infinite unless the source of real world data is, and in such a case the clustering process would then also have to be incremental. Moreover, access to such an extensive and reliable source of real-world data streams is still necessary and domain knowledge is still necessary to analyse concept drift. Finally — this problem would be much more suited to a multi-label *ranking* problem, as opposed to classification. Hence we do not consider this idea further.

The task of generating synthetic multi-label data streams has not yet been thoroughly investigated, and has been mainly specific to certain algorithms or scenarios. In following sections, this paper presents a general framework for multi-label synthetic data generation designed to produce a wide variety of multi-label data in the form of a data stream.

3 Generating Multi-label Data Streams

The main novelty of the framework presented in this paper stems from the use of *problem transformation*, also known as *data transformation*, well known in the multi-label literature [8, 6, 9]. It has shown that it is possible to decompose multi-label data into single-label data. The *reverse transformation is also possible*: single-label data can be transformed into multi-label data. This allows for a generalised framework which can generate multi-label synthetic data independently of the actual data-generation process.

Just as problem transformation classification methods use existing single-label classifiers independently of the transformation method, synthetic multi-label generation can be carried out independently of the data generation method. The `MOA` framework[1] [4] provides state-of-the-art functionality for generating single-label synthetic data streams under a variety of schemes, all of which could be used for creating a multi-label stream. The task of composing a realistic multi-label data stream from single-label data is discussed in depth in this section and later the synthetic data generated is evaluated in comparison to real world data.

Figure 1 outlines the overall process to generate a multi-label example. An initial single-label instance is generated according to label skew, and further single-label examples are generated and are added according to the probabilities that they should occur together. Both the feature spaces and label spaces are combined to form a multi-label

[1] `http://www.cs.waikato.ac.nz/~abifet/MOA/`

example (X, S). All processes, including the combination of feature spaces $(X \oplus X')$ will be described in this section.

```
GENERATEML()
 1  ▷ Generate and filter a single-label example according to skew
 2  $(X, l) \leftarrow filter(Pr(l), \text{SL.GENERATESL}())$
 3  ▷ Formation of a multi-label example
 4  $(X, S \leftarrow \{l\})$
 5  ▷ Adding multiple labels $l'$ where $l' \notin S$
 6  while $|S| < \beta$
 7      do
 8          ▷ Generate and filter a single-label example
 9          $(X', l') \leftarrow filter(Pr(l'|S), \text{SL.GENERATESL}())$
10          ▷ Combine the feature set, and add the label
11          $(X, S) = (X \oplus X', S = S \cup l')$
12  ▷ Hence new multi-label example: $(X, S)$
13  return $(X, S)$
```

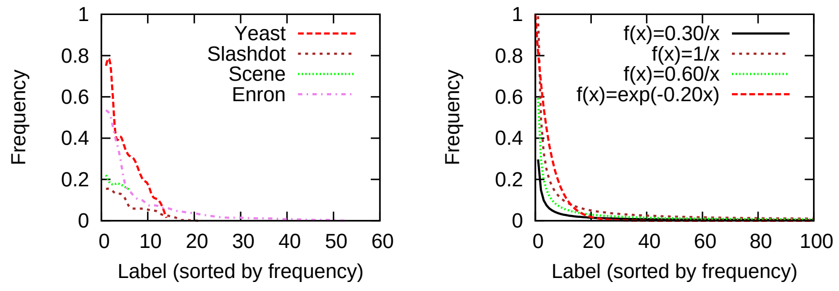
Fig. 1. Generating a multi-label example. `SL.generateSL()` represents any single-label data stream generator from (for example the `MOA` framework). $filter(\gamma, D)$ filters instances from a single-label stream D according to γ, and β is a constant to help approximate a certain label cardinality.

3.1 Label Skew

The phenomenon of *label skew*, where a label or subset of labels are particularly dominant or subordinate in the data, is not unique to multi-label data, but does tend to be particularly prevalent and exaggerated. This is due to the nature of multi-label data: each example can be associated with multiple labels and it is therefore inherently possible for more than one label to dominate the majority of examples (unlike single-label data). Skew in multi-label data is often intuitive, especially to text classification. A label such as `Economy` is likely to be relevant to many examples in a news articles corpus. It is also likely to be found in combination with other labels, for example {`Economy,Politics`}, or {`Economy,New Zealand`}. Therefore `Economy` is likely to be very frequent in the data, while other labels, such as `New Zealand`, only refer to specific subset of news articles therefore occur much more infrequently.

Although label skew is naturally exaggerated in multi-label data by the process of adding multiple labels to single-label data, for the purpose of introducing and controlling concept drift (addressed below), finer grained control over this skew is necessary. Exponential or asymptotic distributions can be used to determine frequencies over a random ordering of the class labels. For example, $f(j) = \frac{\alpha}{j}$, where $f(j)$ represents the frequency of the jth label for some constant α.

New single-label examples can be filtered according to this distribution and the label skew of a data stream D can easily be manipulated by changing α or $f(j)$. When a dataset's label skew is ordered and plotted, a visual representation is obtained. Figure 2 displays the label skews of some real multi-label datasets and functions which approximate them.



(a) Real-world datasets skew (b) Function approximations

Fig. 2. Label skew for various datasets (a) and some function approximations (b). Labels (x axis) sorted by frequency (y axis).

3.2 Label Distribution

The most fundamental difference between multi-label data and single-label data is that instances may be associated with multiple labels, as opposed to a single class label. A multi-label dataset has an average number of labels assigned per example. The average number of labels per example is the *label cardinality*:

$$LC(D) = \frac{1}{|D|}\sum_{i=1}^{|D|}|S_i|$$

Label distribution refers to the overall *composition of the label cardinality*: the frequency of label set sizes in the dataset. This can also be represented as a function $g(i) = n$ where n percent of instances have i labels assigned to them.

A general distinction can be made between two types of label distribution:

Type A : Most examples contain a single label. This is typical of many text and media classification scenarios where most examples fit naturally under a single label scheme, but multi-labelling has been introduced to resolve classification ambiguities. This is the most common type of multi-label data.

Type B : Most examples contain more than one label. The label set is usually very domain-dependent and chosen specifically to represent a multi-label scheme.

Examples of *Type A* include news articles, and media such as images and video. Most images, for example, may fit naturally into a single-label scheme and may have labels such as `Mountains`, `Forest`, or `Sea`. Multiple labels are used to resolve occasional ambiguities such as when `Mountains` and `Forest` are both relevant to one particular image. A good real-world example of this is the *Scene* dataset[2].

Examples of *Type B* include biological datasets where genes are expected to have multiple functions and text datasets like the *Enron* dataset. *Enron* originates from an e-mail corpus[3] and this version[4] of the dataset contains categories which almost take the form of a checklist and were obviously conceived with consideration for multi-label representation. A small subset of *Enron*'s label set is shown in Figure 3 to illustrate a *Type B* labelling scheme.

Label	Note
Attachment(s)	The e-mail contains attachment
Forwarded	The e-mail was forwarded
Legal Advice	The e-mail contains legal advice
Humor	Written with a tone of humour
Admiration	Written with a tone of admiration
...	...

Fig. 3. An example (from *Enron*) of *Type B* label distribution

The label distribution of both types approximates a Poisson distribution (Equation 1). Values of k and λ depend on the data type. *Type A*'s distribution can be approximated $0, POISS(k = \{0, 1, \cdots, |L|\}, \lambda \approx 0.25)$. *Type B*'s distribution can be approximated by $0, POISS(k = \{1, \cdots, |L|\}, \lambda = LC(D))$ (in the latter case $k = 1$ initially, and $LC(D)$ is as defined above).

$$POISS(k, \lambda) = \frac{\lambda^k e^{-\lambda}}{k!} \tag{1}$$

Figure 4 shows the label cardinality distributions of real multi-label datasets alongside the Poisson functions which approximate them.

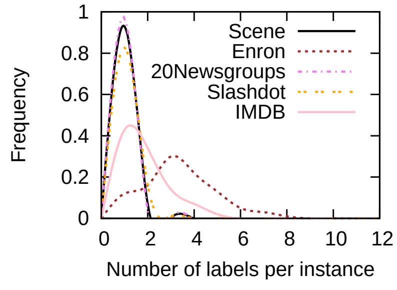
In practice, label cardinality (LC) is never greater than about 5.0. If $LC(D) \gg 5.0$, the problem is usually better treated as a *hierarchical* problem, or *keyword* problem where keywords are *not* assigned based on a predetermined categorical structure intended to facilitate browsing, but rather searching, linking or lookup structures; likewise where $|L| \gg 100$.

Earlier, in Figure 1, a constant β is used to control the assignment of labels so as to adhere to one of the two distributions. This constant is closely linked to the desired label cardinality: i.e. $LC \approx \beta$.

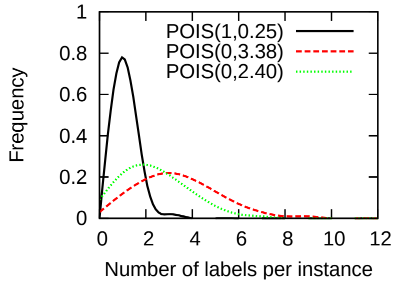
[2] *Scene* can be obtained from `http://mlkd.csd.auth.gr/multilabel.html#Datasets`

[3] `http://www-2.cs.cmu/~enron/`

[4] Using the labelling scheme `http://bailando.sims.berkeley.edu/enron_email.html`, obtainable from `http://www.cs.waikato.ac.nz/~jmr30/#datasets`



(a) Real-world datasets



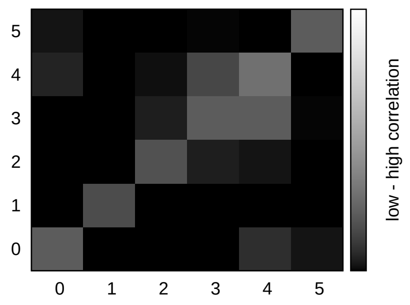
(b) Function approximations

Fig. 4. Label distributions of real-world datasets (a) and approximations (b). The number of labels per example (x axis) against frequency (y axis).

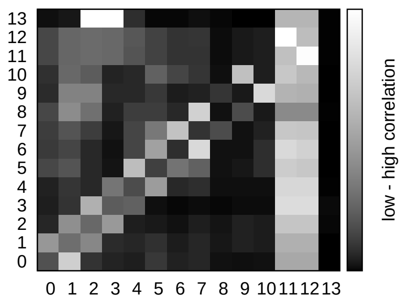
3.3 Label Relationships

There is wide consensus in the literature about the existence of label interdependencies in multi-label data [6, 9, 7, 10]. The underlying relationships between labels in the data is reflective of the problem domain. The degree of label dependency varies, but any real world data in which labels are completely independent of each other is not an interesting multi-label problem, but rather $|L|$ separate binary filtering problems. This implies that labels cannot be selected independently or randomly to create a synthetically generated multi-label example.

Multi-label relationships usually emerge from a problem domain. These relationships can be viewed as a $|L| \times |L|$ *probability matrix* m where $m[k][j] = Pr(l_k|l_j)$. Figure 5 shows matrix representations for the *Scene* (a) and *Yeast* (b) datasets which represent *Type A* and *Type B* data, respectively. The label frequencies are displayed in the matrix diagonal, i.e. $m[k][k] = Pr(l_k)$.



(a) *Scene* dataset (Type A data).



(b) *Yeast* dataset (Type B data).

Fig. 5. Label relationship matrices displayed as heatmaps. The matrix diagonal represents $P(l_i)$ for each $l_i \in L$.

The correlations are related to label skew (covered in Section 3.1). That is to say $Pr(l_j|l_k)$ is high if $Pr(l_j)$ is high, and correspondingly low when $Pr(l_j)$ is low. This is most noticeable in *Yeast* for labels l_{11} and l_{12} where these labels are associated with high frequency and many correlations. Label l_8 and l_{13} show the converse. Other shade differences represent domain dependent factors which can be represented in synthetic data by randomisation. In this particular *Type A* dataset (*Scene*), there is only weak skew and the domain-dependent label correlations stand out clearer.

To generate an artificial matrix m to simulate domain-dependent label correlations, $\varepsilon\%$ of rows under column $m[j]$ are filled with normally-distributed random numbers where $\mu = Pr(l_j)$ and $\sigma = 1.0$ (other cells are left as ≈ 0.0). ε is related to label cardinality and should be set low for *Type A* data (low label cardinality), and high for *Type B* data (higher label cardinality).

3.4 The Feature Space

A complete framework must be able to transform generated single-label examples into multi-label examples, and to do so must consider the *feature space*, and more importantly, the relationship between feature attributes and labels. Text data is both intuitive to examine, and also representative of the majority of multi-label data streams. Tables 1 and 2 show the most frequent words for labels occurring *exclusively* of each other, together *in combination*, and also the *global* most frequent words, for comparison. Figures 6(a) and 6(b) show the Gaussian distributions for specific examples taken from the tables. Slashdot[5] contains summaries of news articles and *20 Newsgroups*[6] contains newsgroup posts.

Referring to these tables and figures, two feature-label effects can be seen which contain information that can benefit a multi-label algorithm:

A *feature-label effect* is where a feature identifies a certain label. An intuitive example is in the *Slashdot* dataset where 'linux' pertains strongly to the label `Linux`, while 'mobile' pertains to `Mobile`, and both words are relevant where these labels are found in combination.

A *feature-combination effect* is where a feature identifies a combination of labels. Often some words may occur frequently only when two labels are found in combination. This is the case in the *20 Newsgroup* dataset for the word 'arms'. This feature is relevant to `politics.guns` but tends to occur even more frequently when the newsgroup post is also posted to `misc.religion`.

There are also various *random effects*. Words like 'anonymous' are generic and do not provide information regarding the presence of labels or combinations of labels. They may occur less frequently in a label combination simply because with an average paragraph length of n words, over several labels, there are fewer words between labels and words which are more strongly relevant (i.e. resulting from the *feature-label effect* and *feature-combination effect*) take preference.

A surprisingly uncommon and irrelevant effect is the average occurrence of two features in a combination: $P(x|\{A,B\}) \approx (P(x|A)+P(x|B))/2$ for feature x and labels

[5] `http://slashdot.com`
[6] `http://people.csail.mit.edu/jrennie/20Newsgroups/`

A,B. This effect can also be considered random because it does not tend to indicate the presence of either a specific label or combinations of labels.

Table 1. *Slashdot*. Most frequent words for labels Linux and Mobile

Global	Linux	Mobile	{Linux,Mobile}
anonymous	linux	mobile	linux
reader	ubuntu	iphone	open
game	source	anonymous	windows
story	open	reader	phone
reports	released	phone	netbook
world	anonymous	android	source
years	kernel	apple	mobile
released	software	phones	free

Table 2. *20 Newsgroups*. Most frequent words for labels politics.guns and religion.misc

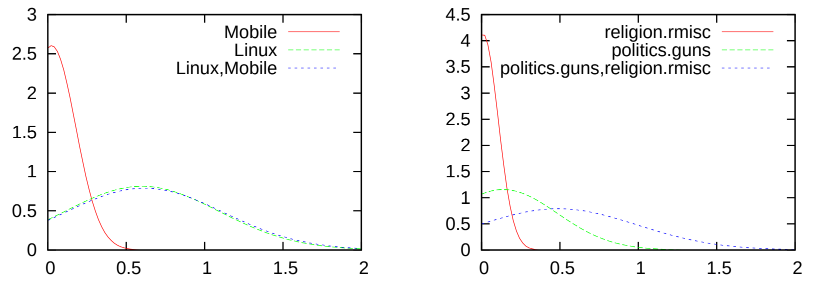
Global	politics.guns	religion.misc	{politics.guns,religion.misc}
don	people	don	jews
1	don	people	arms
2	gun	christian	bear
people	time	god	don
time	government	years	koresh
good	fbi	good	fbi
make	guns	time	people
3	waco	make	news

In implementation, parameters can be used to influence the proportions of the two effects and a mapping is used to carry this out where each feature attribute is mapped to either a single label or a label pair or neither while the remaining proportion implies random effects. Each attribute either implies the presence of a particular label (the *feature-label effect*), implies the presence or absence of a particular combination the (*feature-combination effect*), or does not imply anything (a *random effect*). The process is outlined in Figure 7.

3.5 Concept Drift

It is known that real-world data streams inevitably begin to show changes to the concepts they represent [1]. This known as *concept drift*. If the concept drift is particularly abrupt, it may be called *concept shift*.

In addition to concept drift in the feature space, as found in single-label data streams, multi-label data also involves concept drift to label cardinality, label skew, label distribution, label-label relationships and feature attribute-label relationships. All of these



(a) *Slashdot* for ‘linux’ (b) *20 Newsgroups* for ‘arms’

Fig. 6. Word frequencies for certain labels individually and in combination.

have been discussed above. Some multi-label concept drift may also involve a change to the label set (L) itself. Figures 8(a) and 8(b) show the effect under two measures of shift: “label set coverage” refers to the percentage of instances where label sets overlap the label sets of the initial instances. Label combinations are recorded for the first 100 examples $d_0, \cdots, d_{99}$ and then the percentage of reused combinations is plotted for each of the following blocks of 100 examples: $d_{100}, \cdots, d_{199}, d_{200}, \cdots, d_{299}, \cdots$. This is a form of measuring concept drift in the *label space*. “Accuracy” refers to classification performance under Naive Bayes with a threshold to create multi-label sets (refer to the *ranking and threshold* method in Section 4.1). This is a way to measure concept drift in both the *instance space*, and *feature space*. *Yeast* is a randomised batch dataset, as opposed to a stream, and is displayed for purposes of comparison. In both cases, there is indication of concept drift when the plot is unstable. *20 Newsgroups* shows a very abrupt change in the label space, while *Enron* shows pronounced drift early on. *Slashdot* varies only slightly more than the batch dataset *Yeast*.

Recent work by [1] on single-label data streams models concept drift with a sigmoid function. The sigmoid function in Equation 2 represents concept drift for instances $d_0 \cdots d$. This function is also suitable for creating concept drift in multi-label data, where sigmoid functions are applied to all aspects of multi-label data: the label skew, distribution, and relationship matrix.

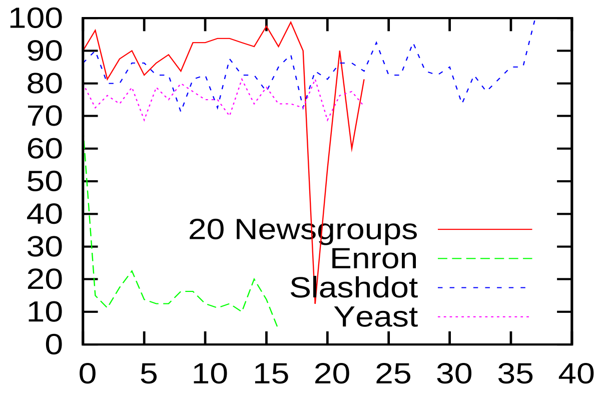
$$sig(d) = \frac{1}{(\Delta x + e^{-s(d-d_0)})} \tag{2}$$

A value x may represent any value of the original concept, and x' the same value in the new concept. To generate a new concept, x' is chosen randomly from a Gaussian distribution where $\mu = x$ and $\sigma = v$ where if $x' < 0 || x' > 1$ then $x' = -x'$, and where v is supplied as a global parameter to control the *extent* of concept drift. Hence the change for a value x is $\Delta x = (x' - x)$.

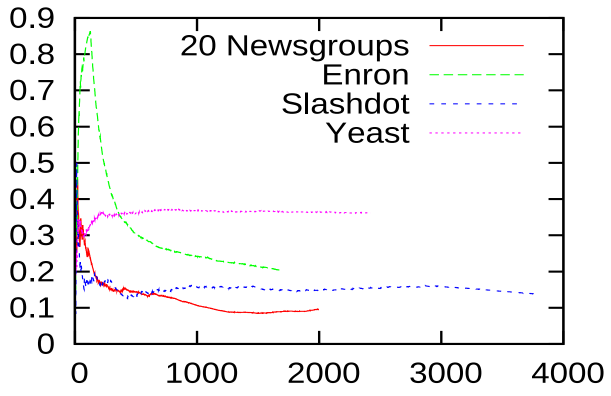
The variable s controls the *abruptness* of the drift. Large values of s create rapid concept drift while smaller values create a more gradual concept drift. The value of s is directly related to the length of change ($d_0 - d$) via a constant e.g. $(d_0 - d)$ = $\frac{s}{8}$.

```
CREATEML()
▷ Begin with ML example of an empty instance $X$ and relevant labels $S$
$(X = (x_1, x_2, \cdots, x_N) \in 0^N, S)$
▷ Generate SL examples to use, one for each label in $S$
$(W_1, l_1), (W_2, l_2), \cdots, (W_{|S|}, l_{|S|}) : l_i \in S$
▷ Generate two binary examples; one positive, one negative
$(V_1, 0), (V_2, 1)$
▷ A mapping of feature attributes to labels or label pairs
$\zeta$
▷ For each feature attribute in the feature space $X$
for $a \leftarrow 1 \ldots |X|$
    do
        ▷ if the attribute maps to a single label
        if $|\zeta[a]| \leq 1$
          then
              ▷ and if that label is relevant to this example
              if $\zeta[a] = l_i : l_i \in S$
                then
                    ▷ Use value from relevant SL example $(W_i, l_i)$
                    $X[a] = W_i[a]$
                else
                    ▷ Use average from all SL examples (random effect)
                    $X[a] \leftarrow$ AVG$(W_1[a], W_2[a], \cdots, W_{|S|}[a])$
              ▷ otherwise, if the attribute maps to a label pair
          else if $|\zeta[a]| = 2$
                then
                    ▷ and if that label pair is relevant to this example
                    if $\zeta[a] \subseteq S$
                      then
                          ▷ Use value from *positive* binary example
                          $X[a] = V_2[a]$
                      else ▷ Use value from *negative* binary example
                          $X[a] = V_1[a]$
        ▷ A ML example with completed feature space and label set
        $(X, S)$
```

Fig. 7. Creating a multi-label example from several single-label and binary examples into a multi-label example. The process can be governed by the mapping ζ to either influence more of either effect and the empty set $\emptyset$ can be used to create a random effect.



(a) Label set coverage over time



(b) Average accuracy over time

Fig. 8. Label set coverage and accuracy measured over time on real-world data sets.

Figure 9 displays sigmoid functions given different values of s and v. Note that in practice, the functions would be centered around $x' - ((x' - x)/2)$ and not 0.5.

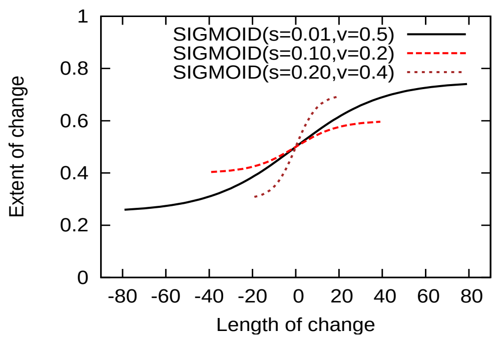


Fig. 9. Sigmoid functions for different changes under different values of s and Δx (v).

4 Results and Discussion

Table 3 shows the range of parameters for generating multi-label data streams under the MOA-based framework. An approximation of label cardinality ($-z$) and a mapping of feature-label relationships ($-a$, $-b$) is sufficient to influence all multi-label dimensions of a dataset.

Table 3. Possible parameters for synthetic data generation.

parameter	type	description	symbol
$-g$	class	single-label generator	SL
$-i$	int	number of instances	$\|D\|$
$-c$	int	number of labels	$\|L\|$
$-u$	int	number of attributes	$\|X\|$
$-r$	int	random seed option	
$-z$	float	desired label cardinality	β, ε
$-a$	float	proportion *Label-Effect* mappings	ζ
$-b$	float	proportion *Combination-Effect* mappings	ζ
$-v$	int	average extent of change	Δ
$-x$	int	length/range of change	$d - d_0$
$-p$	int	beginning point of change	d_0

4.1 Resulting Datasets

In Table 4, statistics are displayed of three real-world multi-label datasets and two synthetic multi-label datasets generated using the framework introduced in this paper.

Synth6 has been designed to approximate *Scene* and *Synth8* is intended to represent a new *Type B* dataset.

Table 4. Statistics relating to real-world and synthetic datasets.

Method	*Scene*	*Synth6*	*Yeast*	*Enron*	*Synth8*
$\|L\|$	6	6	14	53	23
$\|X\|$	300	300	100	1000	500
Type	A	A	B	B	B
Attributes	num.	num.	num	sparse	num.
Label Cardinality	1.07	1.06	4.24	3.38	2.73
Percent Unique	0.006	0.012	0.082	0.442	0.313

Table 5 displays the performance of various standard multi-label base methods on the same datasets as in Table 4. The *majority combination* simply selects the most popular label combination for each test example. The other methods are well known problem transformation methods, all reviewed in [8]. The *label powerset* method treats each multi-label set as a single label, the *binary relevance* method treats each label as a separate binary problem, and the *ranking and threshold* method ranks the relevance of labels to each test example and selects a subset of the highest ranked labels using a threshold to be the multi-label classification set.

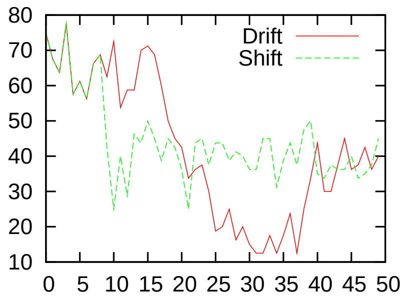
Problem transformation methods require a base single-label classifier to carry out classifications. We use Naive Bayes as the base classifier, which allows incremental classification, even for the label powerset method which requires labels to be added dynamically. The table compares the accuracy of these methods. Accuracy is determined as in [8], but in this case the accuracy is measured for each new example in the stream in a *test then train* scenario.

The variety in the results is to be expected due to the different dimensions of each dataset and each method but, importantly, accuracy is higher than the default method. This means that the method for combining label and feature spaces is creating multi-label relevant information simulative of real-world data.

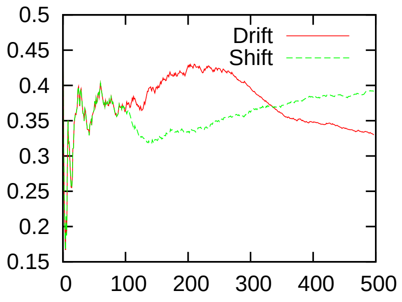
Table 5. The average accuracy of various methods on real and synthetic datasets.

Method	*Scene*	*Synth6*	*Yeast*	*Enron*	*Synth8*
Majority Combination	18.31	18.50	39.05	17.11	20.17
Label Powerset	60.60	34.50	46.63	42.47	37.25
Binary Relevance	46.22	27.50	42.28	18.73	31.42
Ranking and Threshold	65.60	30.25	34.71	23.31	26.37

Finally synthetic concept drift is considered. Figure 10 plots the *label set coverage* 10(a) and *average accuracy* 10(b) over time. Label set coverage varies over the range of the *drift*, before stabilising afterwards. In terms of *shift*, the coverage drops sharply and stabilises. Accuracy decreases suddenly at the beginning of the *shift*, but is able to gradually recover, whereas it declines more slowly over the longer period of the *drift* and is more negatively effected in the long run. This is comparable to the analysis of real-world data earlier in Figure 8.



(a) Label set coverage over time



(b) Average accuracy over time

Fig. 10. Label set coverage and accuracy measured over time on a synthetic data set.

5 Conclusions and Future Work

This paper conducted an in-depth analysis of multi-label data and how the concepts relating to such data change over time in a data stream context. This lead to a framework for generating synthetic multi-label data streams. This framework is based on the concept of problem transformation – it creates a multi-label data stream from a single-label data generator independently of the actual data generation process. It is possible to generate a wide variety of multi-label data by configuring a number of parameters. These parameters allow the manipulation of the multi-label aspects of the data as well as the introduction of concept drift.

Analysis indicates that the data is closely representative of real-world data and therefore able to serve for the analysis and evaluation of incremental multi-label algorithms.

Future work will involve conducting large-scale evaluations of multi-label algorithms using the synthetic multi-label data streams which the framework is capable of generating. This will aid investigations into the multi-label data stream context.

References

1. Albert Bifet and Ricard Gavaldà. Adaptive parameter-free learning from evolving data streams. Technical report, LSI R09-9 Departament de Llenguatges i Sistemes Informàtics Universitat Politècnica de Catalunya, 2009.
2. Lijuan Cai. *Multilabel Classification over Category Taxonomies*. PhD thesis, Department of Computer Science, Brown University, May 2008.
3. Svetlana Kiritchenko. *Hierarchical Text Categorization and its Application to Bioinformatics*. PhD thesis, Queen's University, Kingston, Canada, 2005.
4. Richard Kirkby. *Improving Hoeffding Trees*. PhD thesis, Department of Computer Science, University of Waikato, 2007.
5. Sang-Hyeun Park and Johannes Fürnkranz. Multi-label classification with label constraints. Technical report, Knowledge Engineering Group, TU Darmstadt, 2008.
6. Jesse Read, Bernhard Pfahringer, and Geoff Holmes. Multi-label classification using ensembles of pruned sets. *ICDM '08: IEEE International Conference on Data Mining*, 0:995–1000, 2008.
7. Liang Sun, Shuiwang Ji, and Jieping Ye. Hypergraph spectral learning for multi-label classification. In *KDD '08: 14th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 668–676, New York, NY, USA, 2008. ACM.
8. Grigorios Tsoumakas and Ioannis Katakis. Multi label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3), 2007.
9. Grigorios Tsoumakas and Ioannis P. Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. In *ECML '07: 18th European Conference on Machine Learning*, 2007.
10. Rong Yan, Jelena Tesic, and John R. Smith. Model-shared subspace boosting for multi-label classification. In *KDD '07: 13th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 834–843, New York, NY, USA, 2007. ACM.

Ignoring Co-Occurring Sources in Learning from Multi-Labeled Data Leads to Model Mismatch

Andreas P. Streich and Joachim M. Buhmann

Department of Computer Science, ETH Zurich
Universitätsstrasse 6, CH-8006 Zurich, Switzerland
{andreas.streich, jbuhmann}@inf.ethz.ch

Abstract. The task of multi-label classification is of growing interest in many applications of machine learning. Most currently employed techniques reduce the problem to a series of independent single-label classification problems, thus ignoring the information a data item contains about the other classes it belongs to. Taking a generative viewpoint, we interpret a multi-labeled data item as a combination of independent emissions of all sources it belongs to. We show that if the combination function is a bijection in a single source emission, training an independent generative classifier by maximum likelihood for every class implies a model mismatch.

1 Introduction

Classification is the task of assigning a data point to a set of categories or classes. While it is probably the best studied problem in machine learning, it remains challenging in many cases. A binary classification or dichotomy distinguishes between two classes, while multi-class classification denotes the case of several class choices. Multi-label classification characterizes problems where each data point might belong to more than one class. Instead of a single label, a *label set* is assigned to each data item and indicates all classes this data item belongs to. Typical application areas for multi-labeled classification are acoustic (for example the so-called *Cocktail Party Effect* [1]) and visual scene classification [2], text categorization [7], and bio-informatics [3, 8].

Due to the increasing significance for a large and still rising number of applications, multi-label classification has recently received growing attention. The approaches we are aware of can be grouped into methods that reduce the problem to a (series of) single-label problems, ranking-based methods, algorithm adaptations, and generative models. Current techniques for multi-label classification are reviewed in Section 2.

The previous approaches span a wide variety of algorithms for the multi-label classification problem. Most of them result in a tractable optimization problem and perform reasonably well in some application areas. However, they do not have a clear theoretical background with explicit statements about the underlying assumptions of the algorithm. More concretely, we criticize two main points: Firstly, there is often no clear statement about the meaning of multiple labels. What is the semantics if a data item belongs to more than one class? Secondly, frequent co-occurrence of labels is often equated with similar source statistics. This assumption is often at most vaguely justifiable and smears out the boundaries between different classes.

We provide a theoretical analysis of inference schemes for multi-labeled data. To do so, we take a generative viewpoint and assume that the label set denotes all sources that have contributed to the generation of the data item at hand. If the label set contains a single element, the data item is a direct observation of the respective source. In contrast, if the label set contains more than one element, the data item is understood as a combination of an emissions of each source in the label set. The formal description of the generative model is given in Section 3. In Section 4.1, we define the type of inference schemes which independently estimate parameters for each class based on all data belonging to the class. We show in Section 4 that such inference procedures yield biased estimators and discuss implications of this result in Section 5.

2 Algorithms for Multi-Label Classification

Multi-label classification has attracted a increasing research interest from several application domains within the last two decades. Probably the most basic approach to handle a multi-label classification task is to reduce it to a single-label classification. This can be achieved by considering each unique set of labels found in the training data as one of the classes of a new single-label classification task, as it is done by the model $\mathcal{M}_{new}$. Alternatively, $\mathcal{M}_{ignore}$ ignores training-data with multiple labels. While these techniques, both described in [2], are conceptually very simple, they suffer from sparse training data in most of the practical applications.

Several adaptations of instance-based classifiers for multi-label classification were proposed in the literature. A modified entropy formula was employed in [3] to adapt the C4.5 algorithm for knowledge discovery in multi-label phenotype data. We refer to this method as $\mathcal{M}_{4.5}$. Given the modified entropy formula, frequently co-occurring classes are distinguished only on the bottom of the decision tree. The k-nearest neighbor algorithm was adapted to multi-label data ($\mathcal{M}_{kNN}$) in [16]. Based on the k nearest neighbors, a decision on the class membership of a new data item is taken independently for each class.

Boosting was applied to multi-label text classification e.g. in [12]. Weak learners were trained to either minimize the Hamming loss (AdaBoost.MH) or the ranking loss (AdaBoost.MR). [4] introduced kernel methods for dichotomy learning and ranking in order to solve the multi-label problem.

A technique which is used very often is the so-called *cross-training* or *One-Against-All*, which we betoken with $\mathcal{M}_{cross}$. In a setting where each data item might belong to any subset of K classes, the multi-label problem is replaced by a set of K independent dichotomies: For each class, a classifier determines whether the data at hand belongs to this class. The outputs of the K classifiers are then combined to the final label set. This technique was successfully applied to scene classification [2] and to classify emotions in music [9].

Note that in the usual cross-training, the total weight of a data item is proportional to the number of classes it belongs to. A variation of cross-training is proposed in [11] for improved functional prediction of proteins. The total weight of all data items are equal, the individual weights of labels are either given along with the label set, or, in the absence of such information, are computed by uniformly distributing the total weight over all labels in the label set. We refer to this method as *probabilistic cross-training* $\mathcal{M}_{prob}$.

The technique of label ranking by pairwise comparison [6] can be seen as an extension of cross-training: For any pair of labels (λ_1, λ_2), a classifier is trained to decide which of the two labels are more relevant for a given data item. The $K \cdot (K-1)/2$ preferences are combined to a relevance ranking over all labels. The label set of the data item contains the top-ranked labels, where a "neutral" calibration label is introduced to separate relevant from non-relevant labels [5]. We denote this method by $\mathcal{M}_{CLRPC}$.

All aforementioned techniques assume — be it implicitly or explicitly — that all labels in a label set are mutually independent, also given the training data. We claim that this assumption is often not or only vaguely justified. The principle of maximum entropy is applied in [17] to estimate the dependencies between classes. However, only dependencies in the co-occurrence statistics of labels are considered, while the likelihood of a single label given the data item is still computed independently for each label.

The techniques mentioned so far yield an efficiently solvable optimization problem and perform reasonably well in many practical applications. Yet none of these methods provide a semantics for multi-labeled data. This lack is not only a theoretical inelegance, but gives away domain knowledge which might support the classifier. For the application of text classification, a mixture model ($\mathcal{M}_{mix}$) for the word distribution was presented in [10]. While several sources (representing different topics treated in the document) contribute to the text, each word comes from a single topic source. In the field of acoustics, the parallel emissions of several sources are superposed to the received signal. A deconvolutive approach ($\mathcal{M}_{deconv}$) to handle this situation was presented in [13].

A more detailed review of multi-label classification can be found in [14].

3 Generative Model for Multi-Labeled Data

We take a generative viewpoint and assume that all data is generated by a set $\mathcal{S}$ of K sources. For convenience, we assume $\mathcal{S} = \{1, \ldots, S\}$. Single label data is an observation of one source, whereas multi-labeled data is the combination of several independent source emissions.

More technically, for a data item x, the label set $\mathcal{L} = \{s_1, \ldots, s_d\} \subseteq \mathcal{S}$ denotes the set of sources involved in the generation of x. Each single element s_i, $1 \leq i \leq d$, of $\mathcal{L}$ is called a *label* of x. d is the cardinality of the label set of x. If $d = 1$, x is termed a *single label* data item, whereas we call x a *multi-label* data item if $d \geq 2$. The special case of $d = 2$ is called *binary label*. The set of all possible label sets is denoted by $\mathbb{L}$.

We assume a generative model where each label s corresponds to a source with distribution P_s. All sources are assumed to have the same sample space Ω (If the sample spaces would differ, it might be possible to rule out a certain set of sources for a given

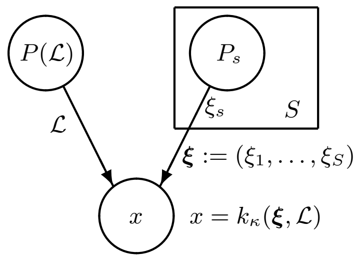

$P(\mathcal{L})$
P_s
$\mathcal{L}$
ξ_s
S
$\boldsymbol{\xi} := (\xi_1, \ldots, \xi_S)$
x
$x = k_\kappa(\boldsymbol{\xi}, \mathcal{L})$

Fig. 1. The generative model for multi-labeled data. An independent sample ξ_s is drawn from each source s. The label set $\mathcal{L}$ is sampled from the label set distribution $P(\mathcal{L})$. These samples are then combined to observation x by the combination function $k_\kappa(\boldsymbol{\xi}, \mathcal{L})$. Note that the observation x does only depend on emissions from sources contained in the label set $\mathcal{L}$.

item without further consideration of any model). The generative process for a data item x with label set $\mathcal{L} = \{s_1, \ldots, s_d\}$ of degree d is illustrated in Figure 1. It consists of three steps:

1. Draw a label set $\mathcal{L}$ from the distribution $P(\mathcal{L})$. Set $d := |\mathcal{L}|$.
2. For each $s \in \mathcal{S}$, draw an independent sample ξ_s from source s.
3. Combine the source samples $(\xi_1, \ldots, \xi_S) =: \boldsymbol{\xi}$ to obtain the observation x. The combination is described by the d-ary combination function k:

$$k_\kappa : \Omega^S \times \mathbb{L} \to \Omega \ ,$$

   where κ is a set of parameters the combination function might depend on. Note that the combination function $k_\kappa(\cdot, \mathcal{L})$ only depends on vector components ξ_s which are in the label set $\mathcal{L}$ and is independent of vector components that are not contained in the label set.

We assume the combination function to be fixed for all data items. Single-label data items are understood as direct observations of a sample from the respective source, i.e. $k_\kappa(\boldsymbol{\xi}, \{s\}) = \xi_s$. We assume parametric distributions P_s, where $\theta_s = (\theta_s^{(1)}, \ldots, \theta_s^{(Q_s)})$ is the tuple of parameters for source s. Q_s is the number of parameters of the distribution P_s. $\Theta = (\theta_1, \ldots, \theta_S)$ denotes the tuple of all parameters of all sources. We assume that these parameters are identifiable, i.e. that there is a one-to-one correspondence between the parameters and the distribution.

With these definitions, the likelihood of the parameters Θ given the data item $D = (x, \mathcal{L})$ is given by

$$L(\Theta; D) = P(\mathcal{L}) \cdot \int \prod_{s \in \mathcal{S}} p_s(\xi_s | \theta_s) \cdot \delta_{k_\kappa(\boldsymbol{\xi}, \mathcal{L}) = x} \, \mathrm{d}\boldsymbol{\xi} \ . \tag{1}$$

For convenience, we define the log-likelihood as $\ell(\Theta; D) := \log L(\Theta; D)$.

We assume samples to be i.i.d. given their source. The probability of a data set $\mathbf{D} = (D_1, \ldots, D_N)$ is thus the product of the probabilities of the data items given their label set. The likelihood of the parameters Θ given the data set $\mathbf{D}$, $L(\Theta; \mathbf{D})$, and the corresponding log-likelihood, $\ell(\Theta; \mathbf{D})$, are given by

$$L(\Theta; \mathbf{D}) = \prod_{n=1}^{N} L(\Theta; D_n) \qquad \ell(\Theta; \mathbf{D}) = \sum_{n=1}^{N} \ell(\Theta; D_n) \; . \tag{2}$$

We propose maximum likelihood estimation, where the parameters are determined such that the probability of the data given the source(s) is maximized:

Definition 1. *(**Maximum-Likelihood estimator**) The estimator $\hat{\Theta}$ for parameter Θ is a* maximum likelihood estimator *if it is determined by*

$$\hat{\Theta} = \arg\max_{\Theta} L(\Theta; \mathbf{D}) \tag{3}$$

where $L(\Theta; \mathbf{D})$ is defined by Equation 2.

While the direct maximization of the likelihood is often difficult, the parameters can equivalently be estimated by maximizing the log-likelihood.

4 Model Mismatch by Ignoring Co-Occurrence

In this section, we present our main contribution. In order to simplify notation, we restrict the main proof to single-label and binary-label data. We first define the type of co-occurrence ignoring inference procedures in Section 4.1. After a few auxiliary lemmata in Section 4.3, we are then ready to give the main theorem for binary labels in Section 4.4. The extension to label sets of any degree is discussed in Section 4.5.

4.1 Co-Occurrence-Ignoring Inference Procedures

In this section, we define co-occurrence-ignoring inference procedures. This class of inference procedures will afterwards be proved to lead to model mismatch.

First, we define when two sources are co-occurring and the notion of partial independence and partial conditional independence:

Definition 2. *(**Co-Occurring Sources**) Two sources s_1, s_2, $s_1 \neq s_2$, are called* co-occurring *in the data set $\mathbf{D}$ if there exists at least one label set in $\mathbf{D}$ containing both s_1 and s_2.*

Definition 3. *(**Partially (Conditionally) Uncorrelated**) Two vectors of random variables $X = (X_1, \ldots, X_{Q_X})$ and $Y = (Y_1, \ldots, Y_{Q_Y})$ are called* partially uncorrelated, *denoted by $X \perp\!\!\!\perp_\partial Y$, if there exist at least one pair of vector components X_{q_1}, Y_{q_2}, $1 \leq q_1 \leq Q_X$ and $1 \leq q_2 \leq Q_Y$, which are uncorrelated, i.e. $P(X_{q_1}, Y_{q_2}) = P(Y_{q_2}) \cdot P(X_{q_1})$.*
If there exists at least one pair of vector components X_{q_1}, Y_{q_2}, $1 \leq q_1 \leq Q_X$ and $1 \leq q_2 \leq Q_Y$, which are uncorrelated conditioned on $\mathbf{D}$ (i.e. $P(X_{q_1}, Y_{q_2}|\mathbf{D}) = P(Y_{q_2}|\mathbf{D}) \cdot P(X_{q_1}|\mathbf{D})$), then X, Y are called partially conditionally uncorrelated, *denoted by $X \perp\!\!\!\perp_\partial Y|\mathbf{D}$.*

In the remainder of this paper, we will focus on the training or inference phase of the classifier. Using a particular inference scheme, parameter estimates are computed based on a data set with corresponding labels. The parameters inferred by the inference scheme $\mathcal{I}$ based on data set $\mathbf{D}$ will be denoted by $\hat{\Theta} = \mathcal{I}(\mathbf{D})$.

Definition 4. ***(Co-Occurrence Ignoring Inference Procedures)*** *An inference procedure $\mathcal{I}$ based on maximum-likelihood is called* co-occurrence ignoring *on a training data set $\mathbf{D}$ if it fulfills the three following conditions:*

1. *$\mathcal{I}$ handles multi-labeled data.*
2. *The likelihood of the parameters θ_s of source s depends only on data items which contain s in their label sets.*
3. *The parameters θ_{s_1}, θ_{s_2} of two co-occurring sources s_1, s_2 are assumed to be partially uncorrelated given the training data $\mathbf{D}$:*

$$\theta_{s_1} \perp\!\!\!\perp_{\partial} \theta_{s_2} | \mathbf{D} \qquad \forall\, s_1, s_2 \in \mathcal{S}.$$

4.2 Model Mismatch

Generative models — be it implicitly or explicitly — model the distribution of both data items and its label sets. Since the training set has only finite size, inference procedures typically suffer from an estimation error, which decreases as more examples are obtained. If the estimated distribution deviates from the true distribution even in the asymptotic case of infinite training data, the inference methods is said to lead yield biased parameter estimators.

There are (at least) two possible causes for biased parameter estimators: First, the inference method itself may be biased. For example, the sample variance is a biased estimator of the variance of a Gaussian distribution. Secondly, a bias in the parameter estimators can be the result of a mismatch between the model assumed by the inference procedure and the true model that generated the data.

Since maximum likelihood estimators are unbiased [15], the first possible cause can be ruled out. We will show that co-occurrence ignoring inference schemes imply a model mismatch and thus cause biased parameter estimators. Note that for identifiable probability distributions, a difference between the estimated and the true parameters implies a difference between the estimated and the true probability distribution.

In later sections, we will rely on representations of the density and the combination function as infinite Taylor series. Functions which can be represented as (infinite) Taylor series are called analytic:

Definition 5. ***(Analytic Function)*** *An* analytic function *is an infinitely differentiable function f on Ω such that the Taylor series at any point $x_0 \in \Omega$, $T(f, x_0, x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n$, converges to $f(x)$ for x in a neighborhood of x_0.*

Polynomials and the exponential function are analytic. Sums, products and compositions of analytic functions are again analytic. Furthermore, the reciprocal of an analytic function that is nowhere zero is analytic.

4.3 Auxiliary Lemmata

In this section, we present four auxiliary lemmata, which we will refer to in the proof of the main theorem. All proofs are given in the appendix.

First, we need to define some notation: Denote by s_1 and s_2, $s_1 \neq s_2$, two sources, parameterized by θ_1 and θ_2, which are assumed to be partially conditionally independent. We denote by $(\theta_{1,c_1}, \theta_{2,c_2})$ a pair of components of θ_1 and θ_2 which are assumed to be conditionally independent given the training data $\mathbf{D}$. Denote by $k_{\kappa,(s_1,s_2)}(\xi_1, \xi_2) := k_\kappa(\boldsymbol{\xi}, \{s_1, s_2\})$ the combination function for the label set $\{s_1, s_2\}$. Let $c_\kappa(\xi_1, x_n) := k^{-1}_{\kappa,(s_1,s_2)}(\xi_1, \cdot)$ be the inverse of the combination function $k_{\kappa,(s_1,s_2)}(\xi_1, \xi_2)$ with respect to the second argument, and set $c_n(\xi) := c_\kappa(\xi_1, x_n)$. We will assume that $k_{\kappa,(s_1,s_2)}(\xi_1, \xi_2)$ is a bijection in ξ_2. All computations are analogous if $k_{\kappa,(s_1,s_2)}(\xi_1, \xi_2)$ is a bijection in ξ_1.

Derivatives of probability densities with respect to parameters are denoted with an upper dot on the density (we assume the parameter with respect to which the derivative is taken is clear from the context). Derivatives with respect to the random variable are denoted by the degree of the derivation in upper brackets:

$$\dot{p}_i(\xi) := \left.\frac{\partial p_i(\xi)}{\partial \theta_{i,c_i}}\right|_{\theta_{i,c_i}=\hat{\theta}^{ML}_{i,c_i}} \qquad p_i^{(m)}(\xi_i) := \frac{\partial^m p_i(\xi)}{\partial \xi_i^m} \qquad \text{for } i = 1, 2\ .$$

Lemma 1. *Assume independent probability density functions $p_i(\xi_i)$ parameterized by θ_i, for $i = 1, 2$. Then, the derivative of the joint distribution $p_{12}(\cdot)$ with respect to both parameters evaluated at the value of the maximum likelihood estimator $\hat{\theta}_i^{ML}$ of the parameter is zero.*
Formally: $\xi_1 \perp\!\!\!\perp \xi_2 \Longrightarrow \left.\frac{\partial^2 p_{12}(\xi_1,\xi_2|\theta_1,\theta_2)}{\partial\theta_1\partial\theta_2}\right|_{\theta_1=\hat{\theta}_1^{ML},\theta_2=\hat{\theta}_2^{ML}} = 0\ .$

The following lemma allows us to rewrite the independence of two parameters given the data as an equality of two sums:

Lemma 2. *Given a training data set $\mathbf{D}$ generated according to the generative process described in Section 3 with a combination function k_κ being a bijection in the emission of at least one source in the label set. If θ_1 and θ_2, the parameter of sources s_1 and s_2, are learned by maximum likelihood, partial conditional independence of θ_1 and θ_2 given $\mathbf{D}$ implies*

$$\begin{aligned}\sum_n &\frac{\int \dot{p}_1(\xi)\dot{p}_2(c_n(\xi))\,\mathrm{d}\xi \cdot \int p_1(\xi)p_2(c_n(\xi))\,\mathrm{d}\xi}{p(x_n)^2} \\ &= \sum_n \frac{\int p_1(\xi)\dot{p}_2(c_n(\xi))\,\mathrm{d}\xi \cdot \int \dot{p}_1(\xi)p_2(c_n(\xi))\,\mathrm{d}\xi}{p(x_n)^2}\ ,\end{aligned} \tag{4}$$

where $p(x_n) := \int p_1(\xi) \cdot p_2(c_n(\xi))\,\mathrm{d}\xi$. Note that only n with $\mathcal{L}_n = \{s_1, s_2\}$ might have a non-zero contribution to the sum. For n with $\mathcal{L}_n \neq \{s_1, s_2\}$, the contributions on either side are 0.

Informally speaking, the independence assumption between two parameter components implies that the partial derivatives of the data likelihood with respect to the respective parameter components can be distributed without changing the value of the expression: On the left hand side, the partial derivative of p_1 and p_2 stand under the same integral, while they are under different integrals on the right hand side.

The following lemma allows us to write the equality condition implied by the independence assumption as an equality of two Taylor series:

Lemma 3. *Assume $c(\cdot)$ is an analytic function and the density functions $p_i(\cdot)$ are continuously differentiable with respect to their parameters θ_i and analytic functions with respect to the random variables ξ_i, for $i = 1, 2$. Then, Equation 4 can be rewritten as an infinite Taylor series*

$$\sum_{k=0}^{\infty} C^k_{lhs} \cdot \xi^k = \sum_{k=0}^{\infty} C^k_{rhs} \cdot \xi^k \tag{5}$$

with coefficients C^k_{lhs} and C^k_{rhs} given by

$$C^k_\alpha = \sum_n \frac{1}{p(x_n)^2} \sum_{l=0}^{k} \frac{C^l_{\alpha,1}(x_n)}{l!} \cdot \frac{C^{k-l}_{\alpha,2}(x_n)}{(k-l)!} \ , \tag{6}$$

where $\alpha = lhs, rhs$ and

$$C^l_{lhs,1}(x_n) = \sum_{m=0}^{l-1} \binom{l-1}{m} \dot{p}_1^{(l-1-m)}(0) \cdot S_m(\dot{p}_2, n) \tag{7}$$

$$C^l_{lhs,2}(x_n) = \sum_{m=0}^{l-1} \binom{l-1}{m} p_1^{(l-1-m)}(0) \cdot S_m(p_2, n) \tag{8}$$

$$C^l_{rhs,1}(x_n) = \sum_{m=0}^{l-1} \binom{l-1}{m} p_1^{(l-1-m)}(0) \cdot S_m(\dot{p}_2, n) \tag{9}$$

$$C^l_{rhs,2}(x_n) = \sum_{m=0}^{l-1} \binom{l-1}{m} \dot{p}_1^{(l-1-m)}(0) \cdot S_m(p_2, n) \tag{10}$$

for $l \geq 1$. $C^0_{\alpha,i}(x_n)$ denote integration constants. Furthermore,

$$S_m(f, n) = \sum_{t \in T_m} b_m(t, f, n) \tag{11}$$

$$b_m(t, f, n) = \frac{m! \cdot f^{(\sum_{i=1}^m t_i)}(c_n(0))}{\prod_{i=1}^m t_i!} \cdot \prod_{i=1}^{m} \left(\frac{c_n^{(i)}(0)}{i!} \right)^{t_i} \tag{12}$$

$$T_m = \left\{ (t_1, \ldots, t_m) \in \mathbb{N}_0^m | \sum_{i=1}^{m} i \cdot t_i = m \right\} \ . \tag{13}$$

The next lemma shows that the equality condition for the two Taylor series (Eq. 5) implies that all derivatives of $c_n(\cdot)$ evaluated at 0 must be equal to zero, i.e. that $c_n(\cdot)$ is a constant in the neighborhood of 0.

Lemma 4. *Given coefficients C^k_{lhs}, C^k_{rhs} as defined in Equations 6 to 13, Equation 5 implies that all derivatives of $c_n(\xi)$ with respect to ξ evaluated at $\xi = 0$ must be zero, i.e. $c_n^{(i)}(0) = 0 \, \forall i \in \mathbb{N}$.*

4.4 Main Theorem for Binary Labels

Theorem 1. *Given a training data set $\mathbf{D}$ with single-label and binary-label data generated according to the generative process described in Section 3. All source distributions are assumed to be continuously differentiable w.r.t. the parameters, and analytic functions w.r.t. the random variables. The binary combination function is a bijection in the emission of at least one source in the label set, and its inverse with respect to any single argument is an analytic function. Then, any co-occurrence ignoring inference scheme $\mathcal{I}$ trained by maximum likelihood on $\mathbf{D}$ suffers from model mismatch.*

The proof is done by contradiction. We assume that $\mathcal{I}$ finds asymptotically the true parameters and then show that this assumption yields a contradiction.

Proof. Assume the model adopted by $\mathcal{I}$ to explain the data set $\mathbf{D}$ matches the true model of the generative process. As the inference scheme $\mathcal{I}$ is co-occurrence ignoring, there is at least one pair of parameters θ_1, θ_2 which do not parameterize the same source and which $\mathcal{I}$ assumes to be conditionally independent given the training data $\mathbf{D}$. We denote the source distributions parameterized by θ_i by p_i $(i = 1, 2)$. By Lemma 2, conditional independence of parameters θ_1, θ_2 implies that the optimality condition of maximum likelihood (Eq. 3) yields the condition given in Eq. 4.

Since $c_n(\cdot)$ is an analytic function, we can use Lemma 3 to rewrite Equation 4 as an infinite Taylor series as given in Equation 5. By Lemma 4, this implies that all derivatives $c_n^{(i)}(0)$ for $i \geq 1$ are zero. Since $c_n(\cdot)$ is assumed to be analytic, it must be a constant in the neighborhood of 0. This is a contradiction to the assumption that $c_n(\cdot)$ is a bijection. Therefore, the assumption that $\mathcal{I}$ adopts the true model has to be rejected. ⊠

4.5 Extension to Labels of Higher Degree

In order to avoid too much clutter in the notation, we have given the proof only for training data containing single- and binary-labeled data. The following corollary generalized Theorem 1 to combination functions of any arity.

Corollary 1. *Given a training data set $\mathbf{D}$ with single-label and multi-label data of any order generated according to the generative process described in Section 3. All source distributions are assumed to be continuously differentiable w.r.t. the parameters, and analytic functions w.r.t. the random variables. The combination function is a bijection in the emission of at least one source in the label set, and its inverse with respect to any single argument is an analytic function. Then, any co-occurrence ignoring inference scheme trained by maximum likelihood on $\mathbf{D}$ suffers from model mismatch.*

The proof is very similar to the proof given for the case of binary labels. For lack of space, we only give a sketch of the proof in the appendix.

5 Implications for Multi-Label Classification

In the generative model described in Section 3 and depicted in Figure 1, the single label sources are independent, and the observations with multiple labels are combinations of the emissions of these sources. Since the binary combination function is assumed to be a bijection in one argument if the other argument is clamped to a fixed value, this introduces a one-to-one functional dependency between the observation and one of the source samples. It is therefore not surprising that co-occurrence ignoring classifiers incur a model mismatch. In the following, we discuss the implications of the theorem for the performance of different multi-label classifiers.

First of all, instance-based classification schemes like $\mathcal{M}_{kNN}$ and $\mathcal{M}_{4.5}$ do not estimate any distribution parameters. The theorem is therefore not applicable to these classifiers and does not allow to draw any conclusions on their performance.

The definition of co-occurrence ignoring inference schemes (Section 4.1) requires that the inference scheme handles data with multiple labels. Out of the techniques presented in Section 2, $\mathcal{M}_{ignore}$ and $\mathcal{M}_{new}$ do not match this requirement.

The classification methods $\mathcal{M}_{cross}$ and $\mathcal{M}_{prob}$ do handle multi-labeled data and are co-occurrence ignoring as they are described in the mentioned publications. Both methods independently learn source parameters for each class and use also data with multiple labels for this. In doing so, they are disregarding the contributions of classes in the label set other than the currently trained one. This leads to a systematic deviation of the parameter estimators from the true parameter values.

In the pairwise ranking method $\mathcal{M}_{CLRPC}$, a different set of source parameters is learned for each pair of labels. This allows a different parametrization of the same source for different ”partner” labels. This model assumption does not agree with the generative model in Eq. 1 and thus allows a model mismatch.

Both $\mathcal{M}_{mix}$ and $\mathcal{M}_{deconv}$ take into account co-occurring labels. The presented theorem does therefore not imply a mismatch of these two methods with the true source — provided the combination function assumed in the models matches the true combination function, and the true source distributions can be described with the parametric distributions assumed by the model.

With regard to the distributions, the theorem assumes density functions which are continuously differentiable with respect to the parameters and analytic functions with respect to the random variables. Most continuous probability distributions fulfill these requirements. Exceptions are e.g. the Dirac delta function and the Cantor distribution.

The combination function is assumed to be a bijection in one of the arguments. Most elementary mathematical operations like (weighted) sum and difference, product and exponentials as well as combinations thereof are bijections. Softmax is also a bijection in any of the arguments. However, other combination functions like maximum and minimum are not bijections, and the theorem does not allow to draw any conclusion on inference procedures in this case.

Keeping this in mind, we recommend to use generative classifiers like $\mathcal{M}_{mix}$ or $\mathcal{M}_{deconv}$ for multi-label classification whenever the generative process is sufficiently well known and the resulting optimization problem is stable and solvable within reasonable time. If a generative model can not be employed, one might either still use a classifier based on independent pairwise classifications, being aware that a model mismatch is inevitable and might lead to suboptimal classification performance. Alternatively, the problem might be addressed by an instance-based classification technique like $\mathcal{M}_{kNN}$ and $\mathcal{M}_{4.5}$, or using a generative inference procedure which do not rely on any source independence, such as $\mathcal{M}_{ignore}$ or $\mathcal{M}_{new}$. The latter option, however, is not recommended if only if only a small number (compared to the number of label sets) of training data is available, as is the case in most real-world applications.

6 Conclusion and Outlook

Multi-label classification problems are often addressed by reduction to a independent single-label multi-class problems. While this approach is conceptually simple and results in tractable optimization problems, we have shown that such a classifier is inconsistent with a large group of generative models for data generation. It will therefore suffer from model mismatch.

The presented statement is valid for generative classifiers on data generated with a combination function which is a bijection in any of its arguments and which inverse with respect to one argument is an analytic function. The source distributions are assumed to be analytic functions. We conjecture that a similar statement as the one presented in this paper can be made for a larger class of combination functions, and further types of classifiers.

Furthermore, an important task is the quantification of the mismatch. This will allow to compare the error incurred due to model mismatch with other sources of error (e.g. due to unprecise estimators obtained from a small sample set) and thus to choose the most reliable estimation procedure for a given task.

Appendix

Proof of Lemma 1: The independence implies $p_{12}(\xi_1,\xi_2|\theta_1,\theta_2) = p_1(\xi_1|\theta_1)\cdot p_2(\xi_2|\theta_2)$, thus

$$\frac{\partial^2 p_{12}(\xi_1,\xi_2|\theta_1,\theta_2)}{\partial\theta_1\partial\theta_2} = \frac{\partial \dot{p_1}(\xi_1|\theta_1)}{\partial\theta_2} p_2(\xi_2|\theta_2) + \dot{p_1}(\xi_1|\theta_1)\dot{p_2}(\xi_2|\theta_2)\;. \tag{14}$$

Since $p_1(\cdot)$ is independent of θ_2, the first summand is zero. If θ_1 has the maximum likelihood value, $\dot{p_1}(\xi_1|\theta_1) = 0$, and the sum is equal to zero. ⊠

Proof of Lemma 2: Using the notation of Section 4.3, the log-likelihood of a data item $D_n = (x_n, \mathcal{L}_n)$ with binary label set $\mathcal{L}_n = \{s_1, s_2\}$ can be written as

$$\ell(\Theta; D_n) = \log P(\mathcal{L}_n) + \log\left(\int p_1(\xi) p_2(c_\kappa(\xi, x_n))\,\mathrm{d}\xi\right)\;. \tag{15}$$

Recall that the likelihood of the parameter vector θ_1 is assumed to depend only on data items which contain s_1 in their label set, and that sources are assumed to emit i.i.d. samples. Thus, the derivative of the log-likelihood of the data set $\mathbf{D}$ with respect to θ_{1,c_1} can be written as

$$\frac{\partial \ell(\Theta; \mathbf{D})}{\partial \theta_{1,c_1}} = \sum_{\substack{n=1 \\ \mathcal{L}_n=\{s_1\}}}^{N} \frac{\partial \ell(\Theta, D_n)}{\partial \theta_1, c_1} + \sum_{\substack{n=1 \\ \mathcal{L}_n=\{s_1,\nu_n\}}}^{N} \frac{\partial \ell(\Theta, D_n)}{\partial \theta_{1,c_1}} \, . \tag{16}$$

The first summand on the right hand side of Equation 16 accounts for single label data. By the definition of the inference procedure $\mathcal{I}$, the parameter vector θ_2 and the sum are independent. The second term describes the influence of data items with two labels. Again due to the assumption of $\mathcal{I}$, its derivative with respect to θ_{2,c_2} vanishes for all n with $\nu_n \neq s_2$. For the remaining n with $\mathcal{L}_n = \{s_1, s_2\}$, we have, using Lemma 1:

$$\sum_{\substack{i=1 \\ \mathcal{L}_n=\{s_1,s_2\}}}^{N} \frac{\partial}{\partial \theta_{2,c_2}} \left\{ \frac{\partial \ell(\Theta; \mathbf{D})}{\partial \theta_{1,c_1}} \right\} = 0 \, .$$

Deriving $\ell(\Theta; D_n)$ as defined in Eq. 15 with respect to θ_{1,c_1} and interchanging the derivation and the integration yields

$$\sum_{\substack{n=1 \\ \mathcal{L}_n=\{s_1,s_2\}}}^{N} \frac{\partial}{\partial \theta_{2,c_2}} \left\{ \frac{\int \left(\left. \frac{\partial p_1(\xi)}{\partial \theta_{1,c_1}} \right|_{\theta_{1,c_1}=\hat{\theta}_{1,c_1}} \cdot p_2(c_\kappa(\xi, x)) \right) \mathrm{d}\xi}{\int p_1(\xi) p_2(c_\kappa(\xi, x)) \, \mathrm{d}\xi} \right\} = 0 \, .$$

Applying the derivation with respect to θ_{2,c_2} and using the introduced notation, one gets Eq. 4. ⊠

Proof of Lemma 3: The proof mainly consists of computing the Taylor series of all integrands around $\xi = 0$. The resulting polynomials are then integrated and the coefficients reordered.

The Taylor series of a function $f(x)$ around x_0 is defined as

$$T(f, x_0, x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!} (x - x_0)^k \, ,$$

where $f^{(k)}(x_0)$ is the kth derivative of f evaluated at x_0. The kth derivative of a product $f(x) = f_1(x) \cdot f_2(x)$ is given by the *Leibnitz's law* as

$$f^{(k)}(x) = \sum_{j=0}^{k} \binom{k}{j} f_1^{(j)}(x) \cdot f_2^{(k-j)}(x) \, .$$

The generalization of the chain rule for derivatives of higher order is given by *Faà di Bruno's formula*:

$$\frac{\partial^m}{\partial x^m} \left\{ f(c_n(0)) \right\} = S_m(f, n) \, ,$$

with $S_m(f, n)$ defined in Eq. 11. The Taylor series of the four integrands are thus:

$$\begin{aligned}
\dot{p}_1(\xi) \cdot \dot{p}_2(c_n(\xi)) &= \sum_{l=0}^{\infty} \frac{\xi^l}{l!} \sum_{m=0}^{l} \left\{ \binom{l}{m} \dot{p}_1^{(l-m)}(0) \cdot S_m(\dot{p}_2, n) \right\} \\
p_1(\xi) \cdot p_2(c_n(\xi)) &= \sum_{l=0}^{\infty} \frac{\xi^l}{l!} \sum_{m=0}^{l} \left\{ \binom{l}{m} p_1^{(l-m)}(0) \cdot S_m(p_2, n) \right\} \\
p_1(\xi) \cdot \dot{p}_2(c_n(\xi)) &= \sum_{l=0}^{\infty} \frac{\xi^l}{l!} \sum_{m=0}^{l} \left\{ \binom{l}{m} p_1^{(l-m)}(0) \cdot S_m(\dot{p}_2, n) \right\} \\
\dot{p}_1(\xi) \cdot p_2(c_n(\xi)) &= \sum_{l=0}^{\infty} \frac{\xi^l}{l!} \sum_{m=0}^{l} \left\{ \binom{l}{m} \dot{p}_1^{(l-m)}(0) \cdot S_m(p_2, n) \right\} .
\end{aligned} \tag{17}$$

After integrating the two polynomials separately, multiplying and re-arranging terms, we get the expressions presented in the Lemma. ⊠

Proof of Lemma 4: The proof is done by induction over the order of the derivative.
Base case: Equations 6 and 17 show that setting $C^0_{lhs} = C^0_{rhs}$ and $C^1_{lhs} = C^1_{rhs}$ implies that all integration constants are equal to zero (unless we are willing to accept constraints on the probability densities and their derivatives). With this, the non-zero terms of C^k_{lhs} and C^k_{rhs} for $k = 2, 3$ are identical and thus do not allow to draw any conclusions on the value of the derivatives of the inverse combination function $c_n(\cdot)$. $C^4_{lhs} - C^4_{rhs}$ is the first non-vanishing difference between the coefficients that contains a derivative of $c_n(\cdot)$:

$$\begin{aligned}
C^4_{lhs} - C^4_{rhs} = \frac{1}{12} \cdot c^{(1)}(0) \cdot \Big(\dot{p}_1(0) \cdot p_1^{(1)}(0) - p_1(0) \cdot \dot{p}_1^{(1)}(0) \Big) \\
\cdot \sum_n \frac{\dot{p}_2(c_n(0)) \cdot p_2^{(1)}(c_n(0)) - \dot{p}_2^{(1)}(c_n(0)) \cdot p_2(c_n(0))}{p(x_n)^2} .
\end{aligned} \tag{18}$$

Requiring the left-hand side to be zero implies that at least one factor on the right-hand side has to be zero. Requiring one of the two factors containing probability densities to be zero might be impossible or at least a very hard constraint on the family of distributions and its parameters[1]. The only factor in Equation 18 which is independent of probability densities is $c^{(1)}(0)$. Therefore $C^4_{lhs} = C^4_{rhs}$ implies $c^{(1)}(0) = 0$.
Inductive Step: Assume that, for some integer $z \geq 1$, we have used the conditions $C^{k+3}_{lhs} = C^{k+3}_{rhs}$ for $k = 1, \ldots, z$ to derive $c^{(1)}(0) = \ldots = c^{(z)}(0) = 0$. Using this inductive claim, we show that $c^{(z+1)}(0) = 0$ follows from the condition $C^{z+3+1}_{lhs} = C^{z+3+1}_{rhs}$.

The expression for C^{z+4}_{lhs} can be derived from Eq. 6 with $\alpha = lhs$ and $k = z + 4$. Consider $C^l_{lhs,1}$ as defined in Eq. 7: Since $c^{(i)}(0) = 0$ for $i \leq z$ by the induction claim,

[1] Consider e.g. the univariate Gaussian distributions $p_1(z) = \mathcal{N}(\mu_1, \sigma_1^2)$. With $\dot{p}_1(z) = p_1(z) \cdot \frac{z-\mu_1}{\sigma_1^2}$, $p_1^{(1)}(z) = -p_1(z) \cdot \frac{z-\mu_1}{\sigma_1^2}$ and $\dot{p}_1^{(1)}(z) = p_1(z) \cdot \frac{(z-\mu_1)^2 - \sigma^2}{\sigma_1^4}$, we get $\dot{p}_1(0) \cdot p_1^{(1)}(0) - p_1(0) \cdot \dot{p}_1^{(1)}(0) = -\left[p_1(z)\right]^2 \frac{1}{\sigma_1^2}$. This expression is different from 0 for all parameters and z. A similar reasoning is applicable for the sum over n.

$b_m(t, \dot{p}_2, n)$ is zero whenever there is a $i \leq z$ with $n_i > 0$. Nonzero contributions to $S_m(\dot{p}_2, n)$ are therefore only possible if either $m = 0$ or $m > z$. If $m = 0$, we have $b_0(t, \dot{p}_2, n) = \dot{p}_2(c_n(0))$. In the second case, $z < m \leq l - 1$ and $l \leq z + 3$ limit the possible values of m to $m = z + 1$, $m = z + 2$ and $m = z + 3$. Hence, the only $t \in T_m$ leading to a nonzero contributions are $t_m = 1$ and $t_i = 0$ for all $i \neq m$. Therefore,

$$S_m(p_2, n) = \begin{cases} \dot{p}_2^{(1)}(c_n(0)) \cdot c^{(m)}(0) & \text{if } z < m \leq z + 3 \\ 0 & \text{otherwise.} \end{cases}$$

Now consider $C^l_{lhs,2}$, defined in Eq. 8: With the same argumentation as above, we see that all terms with $0 < m < z + 1$ will become zero. Similar reasonings allow to filter out non-zero contributions to C^{z+4}_{rhs}. Thus, elementary but lengthy calculations and separating different derivatives of $c_n(\cdot)$ lead to:

$$\begin{aligned} C^{z+4}_{lhs} =& \sigma^{z+4}_{lhs} + \sum_n \frac{1}{p(x_n)^2} \Big(\dot{p}_1(0) \cdot \dot{p}_2(c_n(0)) \cdot p_1^{(1)}(0) \cdot p_2^{(1)}(c_n(0)) \\ & \qquad + p_1(0) \cdot p_2(c_n(0)) \cdot \dot{p}_1^{(1)}(0) \cdot \dot{p}_2^{(1)}(c_n(0))\Big) \cdot \frac{z+2}{(z+3)!} \cdot c_n^{(z+1)}(0) \\ & + \sum_n \frac{1}{p(x_n)^2} \Big(\dot{p}_1(0) \cdot \dot{p}_2(c_n(0)) \cdot p_1(0) \cdot p_2^{(1)}(c_n(0)) \\ & \qquad + p_1(0) \cdot p_2(c_n(0)) \cdot \dot{p}_1(0) \cdot \dot{p}_2^{(1)}(c_n(0))\Big) \cdot \frac{1}{(z+3)!} \cdot c_n^{(z+2)}(0) \\ C^{z+4}_{rhs} =& \sigma^{z+4}_{rhs} + \sum_n \frac{1}{p(x_n)^2} \Big(p_1(0) \cdot \dot{p}_2(c_n(0)) \cdot \dot{p}_1^{(1)}(0) \cdot p_2^{(1)}(c_n(0)) \\ & \qquad + \dot{p}_1(0) \cdot p_2(c_n(0)) \cdot p_1^{(1)}(0) \cdot \dot{p}_2^{(1)}(c_n(0))\Big) \cdot \frac{z+2}{(z+3)!} \cdot c_n^{(z+1)}(0) \\ & + \sum_n \frac{1}{p(x_n)^2} \Big(p_1(0) \cdot \dot{p}_2(c_n(0)) \dot{p}_1(0) \cdot p_2^{(1)}(c_n(0)) \\ & \qquad + \dot{p}_1(0) \cdot p_2(c_n(0)) \cdot p_1(0) \cdot \dot{p}_2^{(1)}(c_n(0))\Big) \cdot \frac{1}{(z+3)!} \cdot c_n^{(z+2)}(0) \,. \end{aligned}$$

σ^{z+4}_{lhs} and σ^{z+4}_{rhs} are sums over terms not containing any derivatives of $c_n(\cdot)$. Re-arranging terms and changing summation orders, we get $\sigma^{z+4}_{lhs} = \sigma^{z+4}_{rhs}$.

Finally, the difference $C^{z+4}_{lhs} - C^{z+4}_{rhs}$ is:

$$\begin{aligned} & \frac{z+1}{2 \cdot (z+3)!} \cdot c^{(z+1)}(0) \\ & \quad \cdot \sum_n \frac{1}{p(x_n)^2} \Big(\Big(p_1^{(1)}(0) \cdot \dot{p}_1(0) - p_1(0) \cdot \dot{p}_1^{(1)}(0)\Big) \cdot p_2^{(1)}(c_n(0)) \cdot \dot{p}_2(c_n(0)) \\ & \qquad + \Big(p_1(0) \cdot \dot{p}_1^{(1)}(0) - p_1^{(1)}(0) \cdot \dot{p}_1(0)\Big) \cdot p_2(c_n(0)) \cdot \dot{p}_2^{(1)}(c_n(0))\Big) \,. \end{aligned}$$

As we do not want do put constraints on the source distributions, $c^{(z+1)}(0) = 0$ follows from $C^{z+4}_{lhs} = C^{z+4}_{rhs}$. This proves the induction step and concludes the proof of the Lemma. ⊠

Proof of Corollary 1: The log-likelihood of the parameters Θ given D_n is

$$l(\Theta; D_n) = \log P(\mathcal{L}) + \log \left(\int \prod_{i=1}^{d_n} p_{s_n^{(i)}}(\xi_i | \theta_{s_n^{(i)}}) \cdot \delta_{k_\kappa(\boldsymbol{\xi})=x_n} \, \mathrm{d}\boldsymbol{\xi} \right) ,$$

and the derivative of the parameter likelihood given the training set $\mathbf{D}$ is

$$\frac{\partial \ell(\Theta; \mathbf{D})}{\partial \theta_{1,c_1}} = \sum_{d=1}^{K} \sum_{\substack{n:d_n=d \\ s_1 \in \mathcal{L}_n}} \frac{\partial \ell(\Theta; D_n)}{\partial \theta_{1,c_1}} .$$

Proceeding like in the proof of Lemma 2, we get

$$\begin{aligned} &\sum_{d=1}^{K} \sum_{\substack{n:d_n=d \\ s_1 \in \mathcal{L}_n}} \frac{\int p(\xi_n^{(-1,2)}) \dot{p}_{s_1}(\xi_n^{(1)}) \dot{p}_{s_2}(\xi_n^{(2)}) \, \mathrm{d}\boldsymbol{\xi}_n}{p(x_n)^2} \\ &= \sum_{d=1}^{K} \sum_{\substack{n:d_n=d \\ s_1 \in \mathcal{L}_n}} \frac{\int p(\xi_n^{(-1)}) \dot{p}_{s_1}(\xi_n^{(1)}) \, \mathrm{d}\boldsymbol{\xi}_n p(x_n) \cdot \int p(\xi_n^{(-2)}) \dot{p}_{s_2}(\xi_n^{(2)}) \, \mathrm{d}\boldsymbol{\xi}_n}{p(x_n)^2} , \end{aligned} \tag{19}$$

with the following definitions for a more compact notation:

$$p(\xi_n^{(-1)}) := \prod_{s \in \mathcal{L}_n \setminus \{s_1\}} p_s(\xi_n^{(s)}) \qquad p(\xi_n^{(-2)}) := \prod_{s \in \mathcal{L}_n \setminus \{s_2\}} p_s(\xi_n^{(s)}) \qquad p(\xi_n^{(-1,2)}) := \prod_{s \in \mathcal{L}_n \setminus \{s_1, s_2\}} p_s(\xi_n^{(s)}) .$$

The factors $p(\xi_n^{(-1)})$, $p(\xi_n^{(-2)})$, and $p(\xi_n^{(-1,2)})$ are independent of ξ_1 and ξ_2 and carry through the integration (with respect to ξ_1 and ξ_2) and the Taylor series. The equality of the coefficients of the Taylor series implies again that the combination function is constant with respect to one argument, thus contradicting the assumption that the combination function is a bijection. ⊠

References

1. B. Arons. A review of the cocktail party effect. *Journal of the American Voice I/O Society*, 12:35–50, July 1992.
2. M. Boutell, J. Luo, X. Shen, and C. Brown. Learning multi-label scene classification. *Pattern Recognition*, pages 1757–1771, 2004.
3. A. Clare and R. D. King. Knowledge discovery in multi-label phenotype data. *Lecture Notes in Computer Science*, 2168:42–53, 2001.
4. A. Elisseeff and J. Weston. Kernel methods for multi-labelled classification and categorical regression problems. In *Proceedings of NIPS*, 2002.
5. J. Fürnkranz, E. Hüllermeier, E. L. Mencía, and K. Brinker. Multilabel classification via calibrated label ranking. *Machine Learning*, 73(2):133–153, 2008.
6. E. Hüllermeier, J. Fürnkranz, W. Cheng, and K. Brinker. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172(16-17):1897–1916, 2008.
7. T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of ECML*, 1998.

8. G.R.G. Lanckriet, M. Deng, N. Cristianini, M.I. Jordan, and W.S. Noble. Kernel-based data fusion and its application to protein function prediction in yeast. In *Proceedings of the Pacific Symposium on Biocomputing*, 2004.
9. T. Li and M. Ogihara. Detecting emotion in music. In *Proceedings of the International Symposium on Music Information Retrieval*, pages 239–240, Washington D.C., USA, 2003.
10. A. K. McCallum. Multi-label text classification with a mixture model trained by EM. In *Proceedings of NIPS*, 1999.
11. V. Roth and B. Fischer. Improved functional prediction of proteins by learning kernel combinations in multilabel settings. *BMC Bioinformatics*, 8(S2):S12, 2007.
12. R. E. Schapire and Y. Singer. Boostexter: a boosting-based system for text categorization. In *Machine Learning*, pages 135–168, 2000.
13. A. P. Streich and J. M. Buhmann. Classification of multi-labeled data: A generative approach. In *Proceedings of ECML*, 2008.
14. G. Tsoumakas and I. Katakis. Multi label classification: An Overview. *Int. J. of Data Warehousing and Mining*, 3(3):1–13, 2007.
15. A. van der Vaart. *Asymptotic Statistics*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1998.
16. M.-L. Zhang and Z.-H. Zhou. A k-nearest neighbor based algorithm for multi-label classification. *Proceedings of the IEEE International Conference on Granular Computing*, 2:718–721 Vol. 2, July 2005.
17. S. Zhu, X. Ji, W. Xu, and Y. Gong. Multi-labelled classification using maximum entropy method. In *Proceedings of SIGIR*, 2005.

Correlation-Based Pruning of Stacked Binary Relevance Models for Multi-Label Learning

Grigorios Tsoumakas[1], Anastasios Dimou[2], Eleftherios Spyromitros[1], Vasileios Mezaris[2], Ioannis Kompatsiaris[2], and Ioannis Vlahavas[1]

[1] Dept of Informatics, Aristotle University of Thessaloniki,
Thessaloniki 54124, Greece
{greg,espyromi,vlahavas}@csd.auth.gr
[2] Informatics and Telematics Institute,
6th Km Charilaou-Thermi Rd, Thessaloniki 57001, Greece
{dimou,bmezaris,ikom}@iti.gr

Abstract. Binary relevance (BR) learns a single binary model for each different label of multi-label data. It has linear complexity with respect to the number of labels, but does not take into account label correlations and may fail to accurately predict label combinations and rank labels according to relevance with a new instance. Stacking the models of BR in order to learn a model that associates their output to the true value of each label is a way to alleviate this problem. In this paper we propose the pruning of the models participating in the stacking process, by explicitly measuring the degree of label correlation using the phi coefficient. Exploratory analysis of phi shows that the correlations detected are meaningful and useful. Empirical evaluation of the pruning approach shows that it leads to substantial reduction of the computational cost of stacking and occasional improvements in predictive performance.

1 Introduction

Supervised learning has traditionally focused on the analysis of single-label data, where training examples are associated with a single label λ, from a set of disjoint labels L. However, training examples in several application domains are often associated with a set of labels $Y \subseteq L$. Such data are characterized as multi-label. Though methods for learning from multi-label *textual* data have been proposed since 1999 [14, 19], the years that followed witnessed an increasing number and diversity of applications, such as bioinformatics (e.g. functional genomics) [5, 8, 2, 4, 1, 34], semantic annotation of images [3, 32, 35] and video [17, 20], directed marketing [36], music categorization into genres and emotions [13, 24, 15] and automated tag suggestion in collaborative tagging systems [10, 21].

Binary relevance (BR), one of the most popular multi-label learning methods in the literature, learns a single binary model for each different label of multi-label data independently of the rest of the labels. It has linear complexity with respect to the number of labels and can learn highly optimized (independent parameter optimization process) and potentially specialized (different learning algorithm) binary classifiers for each label using state-of-the-art learning algorithms. In addition, BR can predict arbitrary combinations of labels, without being restricted to those existing in the training set, as is the case for the label powerset algorithm for example [18]. On the other hand, it does not take into account label correlations and may fail to accurately predict label combinations or rank labels according to relevance with a new instance.

One approach that has been proposed in the past in order to deal with the aforementioned problem of BR, works generally as follows: It learns a second (or meta) level of binary models (one for each label) that consider as input the output of all first (or base) level binary models. It will be called 2BR, as it uses the BR method twice, in two consecutive levels. 2BR follows the paradigm of stacked generalization [31], a method for the fusion of heterogeneous classifiers, widely known as stacking.

Variations of 2BR have been successfully used (i.e. achieved improved accuracy compared to BR) by several communities. To the best of our knowledge, it was firstly used by the machine learning and knowledge discovery community in [9], as part of their SVM-HF method, which was based on a support vector machine (SVM) algorithm for training the binary models of both levels. The abstraction of SVM-HF irrespectively of SVMs and its relation to stacking was pointed out in [26, 25]. A very interesting account of the use of 2BR by the image and video processing community is given in Section 1.2 of [17], where it is called context based conceptual fusion. Some of the references therein, precede [9] in date. Finally, 2BR was very recently applied to the analysis of musical titles [15].

As 2BR is based on binary classification models, it retains the aforementioned advantages of BR, apart from the linear time complexity with respect to the number of labels. The number of labels affects both the number of models trained at the meta-level and the dimensionality of their input vector. Another disadvantage of 2BR raises from the fact that some labels can be completely uncorrelated with others. A label that is irrelevant with the one being modeled is not only lacking additional, valuable information for the classification system, but it also introduces extra inherent noise from the base level.

In this paper we propose pruning the base-level models that are considered as input to the meta-level models based on the correlation of labels. The ϕ coefficient is used to calculate the correlation of each label pair based on an initial single pass over the training set. Labels with correlation below a threshold with the label being learned at the meta-level are pruned and the dimensionality of the meta-level feature space is reduced. This approach improves the system efficiency substantially, without significant loss in predictive performance. In some datasets there are even gains in performance due to the reduced noise being introduced to the system.

The rest of this paper is structured as follows. The next section presents the baseline 2BR algorithm, along with the proposed pruning approach. Section 3 describes an exploratory analysis of the distribution and semantics of the ϕ correlation coefficient based on a variety of multi-label data sets. Section 4 describes the setup and results of the empirical evaluation of the proposed approach. Finally, Section 5 concludes and points to interesting extensions of this work for the future.

2 Pruning the 2BR method

This section first gives a formal description of the baseline 2BR method that we adopted in this work. It then motivates the pruning of base-level predictions at the meta-level and presents our approach to achieving it.

2.1 Baseline 2BR

We first provide some notation for the formal description of the algorithm. Let $L = \{\lambda_j : j = 1 \dots M\}$ denote the finite set of labels in a multi-label learning task and $D = \{(\boldsymbol{x_i}, Y_i), i = 1 \dots N\}$ denote a set of multi-label training examples, where $\boldsymbol{x_i}$ is the feature vector and $Y_i \subseteq L$ the set of labels of the i-th example.

2BR accepts as parameters a number of folds F, a base-level binary classification algorithm and a meta-level binary classification algorithm.

The first step of 2BR concerns training the base-level models and gathering their predictions on a set of training examples in order to construct the meta-level training data. One approach here is to use the full training set for both base-level training and prediction gathering [9]. This however can lead to biased meta-level training data. An alternative approach is to hold part of the training set for gathering the predictions. This can lead to a reduced meta-level training set if the original training set is small. This fact was actually one of the two arguments posed against 2BR in [17]. An approach in-between these two was used in [15]: for each label the training was based on a sample of the majority class (typically corresponding to the absence of a label) in order to balance the class distribution.

We follow a different approach here, which makes use of the complete training set for both training and prediction gathering but avoids the biasing problem. Initially, the algorithm splits the training data randomly into F disjoint parts, D_k, $k = 1 \dots F$, of approximately equal size. This process is done separately for each label and independently of the rest of the labels, so that the distribution of each label in each part is similar to its distribution in the complete training set, as in stratified cross-validation. Subsequently, the base-level algorithm is employed $k = 1 \dots F$ times for each label using the set $D \setminus D_k$ for training and the set D_k for evaluation. This process leads to a meta-level training set $D' = \{(\boldsymbol{y_i}, Y_i), i = 1 \dots N\}$, where $\boldsymbol{y_i}$ is a vector containing the predictions of the base-level algorithm for each λ_j given $\boldsymbol{x_i}$. Value y_{ij} denotes the confidence of the algorithm in the annotation of $\boldsymbol{x_i}$ with label λ_j.

The last two steps of 2BR involve: a) training one base-level binary classification model B_j for each label using the base-level algorithm on the complete training set, and b) training one meta-level classification model M_j using the meta-level algorithm on the meta-level training set.

For the classification and/or ranking of a new instance $\boldsymbol{x'}$, first the decision y'_j of each model B_j is obtained. Then the vector of all decisions $\boldsymbol{y'}$ forms a meta-instance, which is given as input to each of the meta-models M_j. Based on the binary output of these models we can obtain a bipartition of the labels (multi-label classification), while if the output is numeric (confidence, probability estimate, etc), then a ranking can also be obtained.

The complexity of 2BR depends on the complexity of the base-level and meta-level algorithms used. If these are given by $f(A,N)$ and $g(A,N)$ respectively for a training set with N examples and A attributes, then the time complexity of 2BR is $O(M\,[Ff(A,N)+g(M,N)])$. The complexity of 2BR with respect to M depends on that of the meta-level learning algorithm with respect to the number of features. For example, if $g(A,N)$ is linear with respect to A, then the complexity of 2BR is quadratic with respect to M.

2.2 Correlation-Based Pruning

In this paper we argue that each meta-level model should not be trained using the predictions of all base-level models. Base-level models corresponding to labels that are not correlated with the label of the given meta-level model should instead be pruned. The motivation is that the higher dimensionality of the input space will only lead to higher cost of training the meta-models, while it might also hurt the generalization of the meta-models.

To achieve our goal, we utilize the ϕ correlation coefficient, a specialized version of the Pearson product moment correlation coefficient for categorical variables with two values, also called dichotomous variables [6]. Given two labels, λ_i and λ_j, and the frequency counts of the combinations of their values given in Table 1, the coefficient is defined as follows:

$$\phi = \frac{AD - BC}{\sqrt{(A+B)(C+D)(A+C)(B+D)}} \tag{1}$$

	λ_j	$\neg\lambda_j$
λ_i	A	B
$\neg\lambda_i$	C	D

Table 1. Contingency table for labels λ_i and λ_j.

The pruned version of 2BR accepts a threshold of ϕ correlation as a parameter, denoted t, where $0 \leq t \leq 1$. When constructing the meta-level training examples for a label λ it only takes into account the predictions of the base-level models for those labels $\lambda' \in L$ whose absolute value of the ϕ correlation with λ is greater or equal to t: $|\phi(\lambda', \lambda)| \geq t$. Obviously, the predictions of the base-level model for λ will always be taken into account, as $\phi(\lambda, \lambda)$ = 1.

If M' is the largest number of base-level models whose predictions are taken into account at the meta-level, then the complexity of the pruned version of 2BR becomes $O(M\,[Ff(A,N)+g(M',N)])$. With appropriate selection of the threshold ϕ, M' can be a much smaller number compared to M, reducing the complexity of 2BR to linear with respect to the number of labels. Alternatively, by explicitly selecting a small number M' of most correlated labels, the linear complexity can be guaranteed.

An approach based on a similar idea, but applied to multiple numerical target variables using decision tree learners, is the Empirical Asymmetric Selective Transfer (EAST) [16]. For each label, EAST performs a greedy forward selection hill climbing search in the space of label subsets, guided by the accuracy of the model. This search process has quadratic complexity with respect to the number of labels. If we also consider the need to train the model at each step, then in the best case this raises the complexity to cubic with respect to the number of labels. Finally, since this process has to be done for all labels, the overall complexity of EAST is quartic with respect to the number of labels. Therefore, EAST is highly inefficient and unsuitable for domains with large numbers of labels. EAST has a clearly different focus (accuracy) compared to our approach (efficiency).

3 Exploratory Analysis of the ϕ Coefficient

This section explores the distribution of the ϕ coefficient in several multi-label data sets in order to derive conclusions on a meaningful range of values for setting the threshold parameter t of the pruned 2BR during the experiments that follow. We also attempt to gain some insight on the semantics underlying the numerical values of the ϕ coefficient by examining the names of the labels in two of these data sets.

3.1 Data sets

We explore the ϕ coefficient on 7 multi-label data sets[3]. Table 2 includes several statistics for each of these data sets [25], including the average number of labels per example (*label cardinality*) and the number of distinct label combinations *distinct labelsets*. Short descriptions of these data sets are given in the following paragraphs.

Table 2. Multi-label data sets and their statistics.

		attributes			**label**	**label**	**distinct**
name	**examples**	**nominal**	**numeric**	**labels**	**cardinality**	**density**	**labelsets**
bibtex	7395	1836	0	159	2.402	0.015	2856
enron	1702	1001	0	53	3.378	0.064	753
mediamill	43907	0	120	101	4.376	0.043	6555
medical	978	1449	0	45	1.245	0.028	94
reuters	6000	0	47236	101	2.880	0.029	1028
tmc2007	28596	49060	0	22	2.158	0.098	1341
yeast	2417	0	103	14	4.237	0.303	198

The *yeast* data set [8] contains micro-array expressions and phylogenetic profiles for 2417 yeast genes. Each gene is annotated with a subset of 14 functional categories (e.g. *metabolism*, *energy*, etc) from the top level of the functional catalogue (FunCat).

[3] Available at `http://mlkd.csd.auth.gr/multilabel.html`

The *tmc2007* data set is based on the data of the competition organized by the text mining workshop of the 7th SIAM international conference on data mining[4]. The original data contained 28596 aviation safety reports in free text form, annotated with one or more out of 22 problem types that appear during flights [22]. Text representation follows the boolean bag-of-words model.

The *medical* data set is based on the data made available during the computational medicine center's 2007 medical natural language processing challenge[5]. It consists of 978 clinical free text reports labeled with one or more out of 45 disease codes.

The *enron* data set is based on a collection of email messages exchanged between the Enron Corporation employees, which was made available during a legal investigation. It contains 1702 email messages that were categorized into 53 topic categories, such as *company strategy*, *humor* and *legal advice*, by the UC Berkeley Enron Email Analysis Project[6].

The *mediamill* data set was part of the Mediamill challenge for automated detection of semantic concepts in 2006 [20]. It contains 43907 video frames annotated with 101 concepts (e.g. *military*, *desert*, *basketball*, etc). The specific dataset we used corresponds to experiment 1 (visual feature extraction) as described in [20]. Each video frame is characterized by a set of 120 visual features.

The *bibtex* data set [10] is based on the data of the ECML/PKDD 2008 discovery challenge. It contains 7395 bibtex entries from the BibSonomy social bookmark and publication sharing system, annotated with a subset of the tags assigned by BibSonomy users (e.g. *statistics*, *quantum*, *datamining*). The title and abstract of bibtex entries were used to construct features using the boolean bag-of-words model.

The *reuters* (rcv1) data set is a well known benchmark for text classification methods. We have used a subset (rcv1subset1) that contains 6000 news articles assigned into one or more out of 101 topics. An extensive description of the rcv1 dataset can be found in [12].

3.2 Mean Label Correlation

Figure 1 depicts a plot of the number of label pairs that exhibit ϕ correlation greater or equal to the corresponding value of the x axis, divided by the total number of labels. We call this number *mean label correlation* as it corresponds to the mean number of correlations of each label that surpass a given threshold of positive or negative ϕ correlation. The plot shows the mean label correlation for the 7 multi-label datasets with respect to a threshold ranging from 0 to 0.3 with a step of 0.01.

Constructing such a plot prior to the execution of 2BR is fast, as it requires a single pass over the data. Based on the calculated correlations of all label pairs, we can choose the threshold parameter t of 2BR, in a way such that a small number of base-level classifiers remains on average for each label, depending on the boost in efficiency that we would like to achieve. The plot shows that after a threshold of 0.3, each label is on average correlated with less than one label (apart from itself) for all datasets.

[4] http://www.cs.utk.edu/tmw07/

[5] http://www.computationalmedicine.org/challenge/

[6] http://bailando.sims.berkeley.edu/enron_email.html

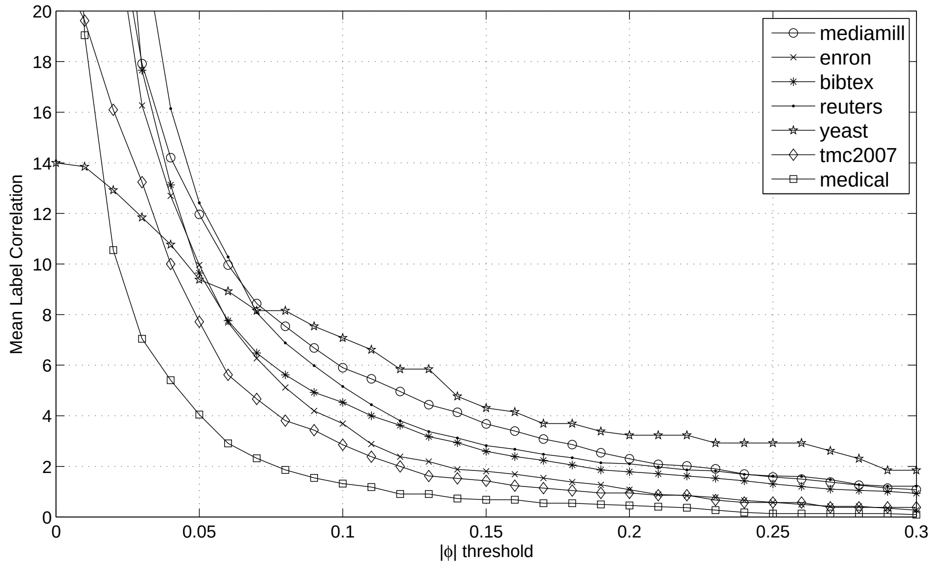


Fig. 1. Mean label correlation with respect to increasing absolute ϕ thresholds in seven multi-label data sets.

Note also that there seems to be a slight correlation between the cardinality of the dataset and the mean number of label correlations per given ϕ threshold. For example, the curves of *yeast* and *mediamill* with a cardinality of around 4 seem to be on top of the rest, despite the difference in number of labels, while on the other side, the curves of *tmc2007* and *medical*, with the smallest cardinality, seem to be lower than the rest of the curves.

3.3 Semantic Analysis of the ϕ Coefficient

The ϕ coefficient of correlation between pairs of labels is estimated from data sets that are typically annotated manually. Manual annotations can be incomplete, erroneous and even biased depending on target and time constraints of the labeling procedure, the accuracy of the labeling effort, and the content of the dataset being labeled. Taking into account all these possible complications, it is useful to examine whether a direct analogy between the ϕ-based and the semantic-based correlation of two labels exists in practice.

In the radar/spider diagram of Figure 2, the ϕ coefficient between two indicative labels, namely *building* and *government leader*, and the rest of the labels from the *mediamill* dataset is depicted. Due to the high number of labels (101), some of them were removed from the diagram to make it more legible. The labels that were removed had zero or close to zero ϕ values with both of the illustrated labels. Moreover, the zero correlation level is highlighted to help the reader distinguish between positive and negative values. It must also be noted that the autocorrelation for both examined labels, equal to 1, has been removed to enhance legibility.

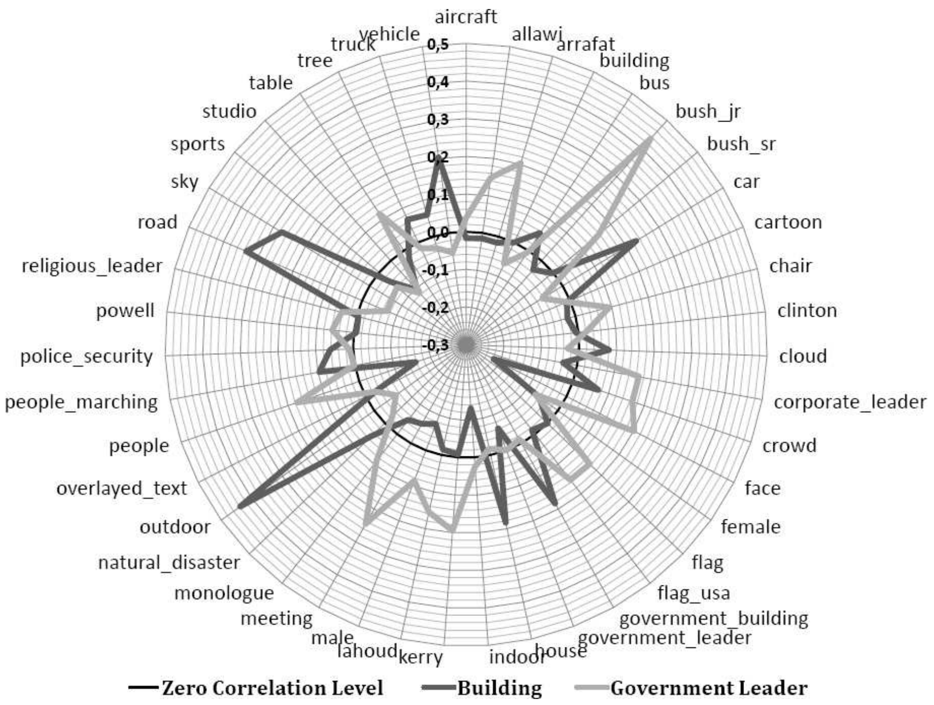


Fig. 2. Radar/spider diagram of the ϕ correlation coefficient of *Building* and *Government Leader* with the rest of the labels.

The label *building* is depicted in the graph with the thick black line. It is positively correlated with the labels *outdoor*, *sky*, *road*, *car*, *truck*, *vehicle*, *tree*, *crowd*, *house*, and *government building*. The labels *house* and *government building* have a direct semantic relation with the label *building*. *outdoor* and *sky* are not part of the concept *building* but they often accompany it in images. This can be explained by the tendency of human annotators to label an image as *building* when the entire building, or at least a good part of it, is depicted. To have such a perspective, the image is taken outdoors and most probably the sky is part of the image. Moreover the concept *building* is strongly related with an urban or suburban environment. In such an environment, the labels *road*, *car*, *truck*, *vehicle*, *tree* and *crowd* are very common. On the other hand, *studio*, *face*, *indoor*, *male*, *government leader*, *meeting* and *people* are labels that do not co-occur with *building* in this dataset. With the exception of label *people* that will be further discussed, all labels are not conceptually compatible with *building*. Label *people* is very similar to *Crowd* and there seems to be an incompatibility. The difference here is in the area of interest. An image depicting a building is not focused on people, making them a crowd without faces and gender. There are also a number of labels like *religious leader*, *overlayed text*, *cartoon*, *Clinton*, *Arrafat*, *sports*, *natural disaster* etc that have zero correlation. These labels show a random relation with the examined one, signaling that the added value that they may offer is very limited if existent.

The label *government leader* is depicted in the graph with the thick grey line. It is strongly and positively correlated with the labels *Allawi*, *Bush Jr*, *Bush Sr*, *Kerry*, *Lahoud*, *Powel*, *chair*, *corporate leader*, *crowd*, *face*, *flag*, *flag USA*, *meeting*, *table*, *people* and *male*. All of them can be mapped to international meetings between leaders and public appearances. Negative ϕ correlation exists with *female*, *studio*, *sports*, *road,* outdoor, *car* and *building*. Judging from the correlations, we can argue that label *government Leader* seems to be biased due to the dataset that is available for the training. It includes mostly American and Arab leaders meeting indoors in an office. Outdoor appearances, female leaders, talks in front of buildings that could also be relevant to the *government leader* label are under-represented in this dataset.

The radar/spider diagram of Figure 3, depicts the ϕ coefficient between two indicative labels, namely *alliances/partnerships* and *company image - changing/influencing*, and the rest of the labels from the *enron* data set. For legibility reasons some labels were removed, and the names of others were shortened or abbreviated in the diagram. All labels that were removed have again zero correlation with the examined labels or a uniform correlation with all the labels.

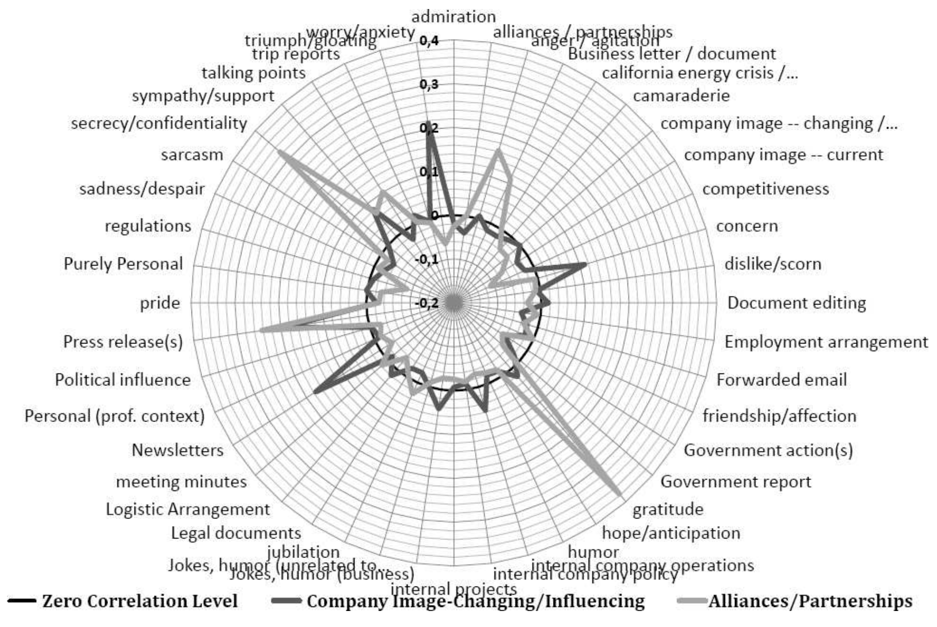


Fig. 3. Radar/spider diagram of the ϕ correlation coefficient of *alliances/partnerships* and *company image - changing influencing* with the rest of the labels.

At this point it is useful to provide some additional background information about the *enron* dataset in order to support the task of the semantic analysis that follows. As discussed in section 3.1, *enron* is based on a collection of e-mail messages. These messages were exchanged between employees of the Enron energy corporation and became available during the legal investigation of a financial scandal, involving Enron and its accounting firm. The data set consists of 53 labels, belonging to 4 main categories, namely *Coarse genre*, *Included/forwarded information*, *Primary topic* and *Emotional tone*. Taking this information into consideration we can explain the inconsistency between some label groups, since they refer to 4 different aspects of an e-mail message. Both the illustrated labels of the radar diagram fall into the *Primary topics* main category.

The label *alliances/partnerships* is depicted in the graph with the thick grey line. It has a strong positive correlation with the labels *gratitude*, *secrecy/confidentiality* and *press release(s)* while a weaker positive correlation emerges with the labels *anger/agitation* and *business letter(s)/document(s)*. An e-mail message with *alliances/partnerships* as the primary topic, is likely to have an emotional tone of *gratitude*, following a successful partnership, or a tone of *secrecy/confidentiality* regarding a prospective alliance, product etc. Furthermore, the reference to *press release(s)* or *business letter(s)/document(s)* in such a message is normal in business practice. On the other hand, a failed partnership can provoke negative emotions, justifying the correlation with the label *anger/agitation*. The most negatively correlated label is *competitiveness/ag-gressiveness*. Again this seems reasonable, since a message labeled as *alliances/ partnerships* is usually lacking aggressive and competitive tones.

The label *company image - changing influencing* is depicted in the graph with the thick black line. It is positively associated with the labels *worry/anxiety*, *press release(s)*, *newsletters*, *concern*, *sympathy/support* and *internal company operations*. For these labels, the conceptual correlation with the changing image of the company is quite straightforward. For example the strong correlation of *worry/anxiety* and *concern* is totally substantiated, considering the darksome future of the company which came in the verge of bankruptcy. On the other hand a negative ϕ correlation is revealed with the labels *government action(s)*, *employment arrangements*, *sarcasm* and *alliance/partnerships*. All these labels are referring to actions and emotions regarding the internal affairs of a company, thus having a negative correlation with *company image*.

Concluding this section we can state that the ϕ coefficient is able to capture both real-life and semantic-based correlations between labels. Furthermore, it is able to point out relations that are not straightforward, e.g. the differences between *people* and *crowd* or the differences between internal and public affairs in a company. Our experiments have shown that in two publicly available datasets with diverse data the relationships mapped are valid and they can prove valuable. Despite its effectiveness, the ϕ coefficient is still dependent on the dataset from which it is extracted and the quality of the annotation.

4 Empirical Evaluation

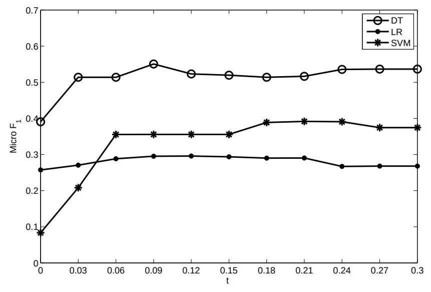
We empirically evaluate the utility of the proposed approach by measuring the performance of the pruned BR^2 on the *yeast* and *enron* data sets, using threshold values ranging from 0 (no pruning) to 0.3 with a step of 0.03. As we saw in the previous section, threshold values greater than 0.3 are not expected to lead to great changes in predictive performance or computational efficiency, as the mean number of correlated labels is already less than two.

In order to be able to draw general conclusions, this comparison should be made with a variety of base and meta-level algorithms. To restrict the large space of experiments, in this paper we focus on decision trees (DT) and linear kernel support vector machines (SVM) for both base-level and meta-level learning. We also use linear regression (LR) at the meta-level, based on the good results for stacking heterogeneous classifiers (i.e. based on different learning algorithms) reported in [23]. In the plots that follow, meta-level DTs, SVMs and LRs are marked with circles, stars and dots respectively.

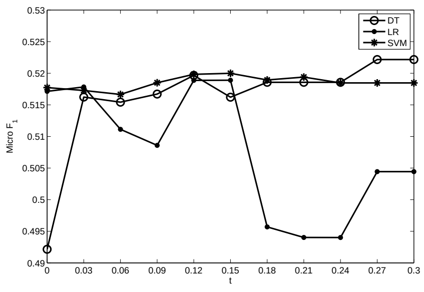
We used the implementations of the above algorithms from the Weka library [30]. We did not perform any parameter optimization for the above algorithms and used them with their default settings, apart from DTs, where Laplace smoothing of the predicted probabilities was enabled. We implemented 2BR by extending the Mulan open source Java library for multi-label learning [29], which works on top of the Weka API.

The performance of 2BR is evaluated in terms of two criteria: a) efficiency, which is measured by the average dimensionality of the meta-level feature vector, and b) accuracy, which is measured using: i) micro F_1, which evaluates bipartitions, and ii) average precision, which evaluates rankings. A description of these and other evaluation measures for multi-label data can be found in [28].

Figure 4 shows the micro F_1 in *enron*. DTs and SVMs are employed as base-level algorithms in sub-figures (a) and (b) respectively. For base-level DTs, pruning seems to increase performance. All meta-level algorithms, namely DT, SVM, and LR, achieve their peak values while employing pruning. Especially for DTs and SVMs, which are the best performing algorithms, the improvement of the F1 measure is substantial. For base-level SVMs, the performance of the meta-level algorithms is either improved or stable with minor variations in terms of the F1 measure.



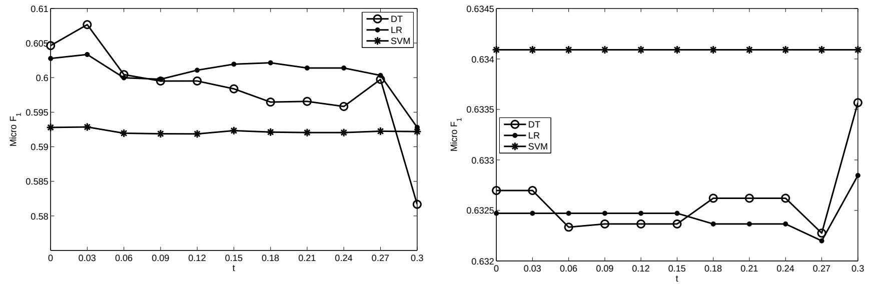
(a) Decision trees.



(b) Support vector machines.

Fig. 4. Micro F_1 in *enron*.

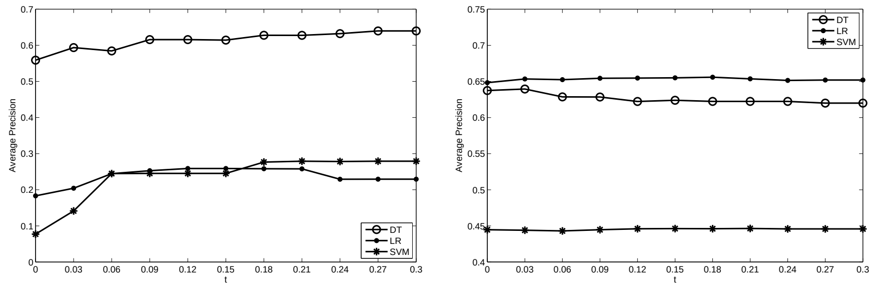
Figure 5 depicts the micro F_1 in *yeast*. The performance trend is the same as in the previous data set. For all combinations of algorithms, the peak value of micro F_1 is obtained with pruning for different thresholds of ϕ. Overall, pruning is enhancing the accuracy of the examined algorithms. Their accuracy is either kept constant, or improved, sometimes substantially.



(a) Decision trees (b) Support vector machines

Fig. 5. Micro F_1 in *yeast*.

In analogy to micro F_1 the average precision is depicted in Figures 6 and 7. In *enron*, pruning is improving the average precision in all cases. Using a DT at both the base and the meta-level gives the best performance in Figure 6(a) showing an almost linear improvement as the ϕ threshold increases. When SVMs are used at the base-level, average precision is almost constant with the peak values obtained always with the aid of pruning.

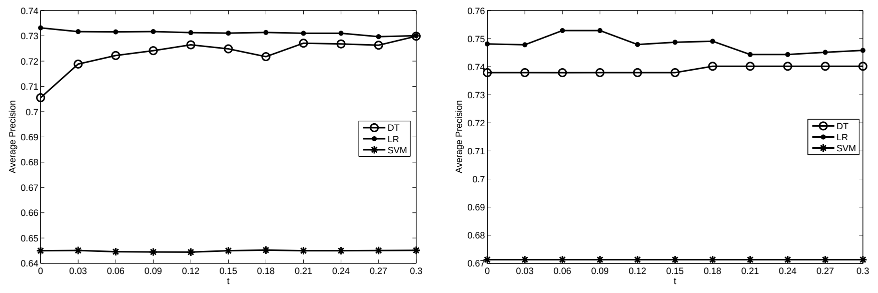


(a) Decision trees (b) Support vector machines

Fig. 6. Average precision in *enron*.

For the *yeast* dataset the conclusions are similar. With the exception of base-level DTs with meta-level LRs, where the performance shows an insignificant decrease, in all other cases pruning seems to improve average precision, including the peak performance. Especially the combination of base and meta-level DT seems to significantly benefit from pruning.

In order to validate the usefulness of pruning we have employed two data sets, three learning algorithms in eight different configurations and two different metrics.



(a) Decision trees (b) Support vector machines

Fig. 7. Average precision in *yeast*.

According to our experiments, the learning procedure seems to benefit from pruning. In several cases there are significant performance improvements, while in all other cases there is no substantial deviation from the accuracy of the baseline 2BR results.

Moreover, the label elimination that takes place, significantly reduces the system complexity, thus increasing its time-efficiency, in all cases where $|\phi| > 0$. Table 3 shows the reduction of the average number of classifiers employed in the meta-level learning for different ϕ thresholds. It suggests that if we bounded the number of labels from the base-level that contribute to the prediction of a label at the meta-level, to a small number (e.g. 5), we could achieve the same or better results at a linear time complexity with respect to M.

	0.03	0.06	0.09	0.12	0.15	0.18	0.21	0.24	0.27	0.30
enron	19.34	10.25	7.08	5.42	4.70	4.21	3.98	3.64	3.34	3.34
yeast	11.57	8.85	8.00	6.29	4.86	4.57	4.00	3.86	3.57	2.86

Table 3. Average number of classifiers being stacked for different ϕ thresholds and data sets.

5 Conclusions and Future Work

In our view, one of the important contributions of this paper is the use of the ϕ coefficient to explicitly quantify the correlation between labels. We believe that this can help improve the scalability and predictive performance of other multi-label methods beyond BR^2 as well. For example, in RAkEL [29], it could be used to construct subsets of correlated labels, with potentially improved performance. It could play the role of the similarity measure in the clustering phase of HOMER [27], perhaps leading to more appropriate clusters. Finally, in DML-kNN [33], it could be utilized in the construction of the margin vectors that are used to characterize the dependency level between labels.

One of the advantages of 2BR, as a classifier fusion method, is that it can encompass additional base-level models at the meta-level. This can be very helpful when the objects to be classified are characterized by different representations (e.g. textual, microarray and clinical descriptors of gene functions) [11, 7]. It can also help with the fusion of heterogeneous base-level classifiers, leading to an improvement of the overall performance. Exploring this direction is among our near future plans, especially since the results of this paper showed that different algorithms work well in different datasets. In both of these scenarios (heterogeneous descriptors and learning algorithms) the pruning is expected to play a more prominent role, as the dimensionality of the meta-level feature vector will grow linearly by a factor equal to the number of different representations/algorithms, unless a hierarchical stacking approach is employed.

We also plan to investigate the relation of pruning to a number of variations of the baseline 2BR algorithm, such as extending the meta-level feature vector with the original base-level features [9, 17] and replacing the numeric meta-level features, which represent the confidence in the binary decision of base-level classifiers, with binary ones, representing the boolean decisions themselves [9, 15].

References

1. Zafer Barutcuoglu, Robert E. Schapire, and Olga G. Troyanskaya. Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7):830–836, 2006.
2. H. Blockeel, L. Schietgat, J. Struyf, S. Dzeroski, and A. Clare. Decision trees for hierarchical multilabel classification: A case study in functional genomics. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4213 LNAI:18–29, 2006.
3. M.R. Boutell, J. Luo, X. Shen, and C.M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771, 2004.
4. Nicolò Cesa-Bianchi, Claudio Gentile, and Luca Zaniboni. Hierarchical classification: combining bayes with svm. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 177–184, 2006.
5. A. Clare and R.D. King. Knowledge discovery in multi-label phenotype data. In *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2001)*, pages 42–53, Freiburg, Germany, 2001.
6. Jacob Cohen, Patricia Cohen, Stephen G. West, and Leona S. Aiken. *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences*. Psychology Press, 2002.
7. A. Dimou, G. Tsoumakas, V. Mezaris, I. Kompatsiaris, and I. Vlahavas. An empirical study of multi-label learning methods for video annotation. In *Proc. 7th International Workshop on Content-Based Multimedia Indexing, CBMI '09*, Chania, Greece, 2009.
8. A. Elisseeff and J. Weston. A kernel method for multi-labelled classification. In *Advances in Neural Information Processing Systems 14*, 2002.
9. S. Godbole and S. Sarawagi. Discriminative methods for multi-labeled classification. In *Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2004)*, pages 22–30, 2004.
10. Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. Multilabel text classification for automated tag suggestion. In *Proceedings of the ECML/PKDD 2008 Discovery Challenge*, Antwerp, Belgium, 2008.
11. Hans-Peter Kriegel, Peer Kröger, Alexey Pryakhin, and Matthias Schubert. Using support vector machines for classifying large sets of multi-represented objects. In *Proc. 4th SIAM Int. Conf. on Data Mining*, pages 102–114, 2004.

12. David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5:361–397, 2004.
13. T. Li and M. Ogihara. Toward intelligent music information retrieval. *IEEE Transactions on Multimedia*, 8(3):564–574, 2006.
14. A. McCallum. Multi-label text classification with a mixture model trained by em. In *Proceedings of the AAAI' 99 Workshop on Text Learning*, 1999.
15. F. Pachet and P. Roy. Improving multilabel analysis of music titles: A large-scale validation of the correction approach. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(2):335–343, 2009.
16. Beau Piccart, Jan Struyf, and Hendrik Blockeel. Empirical asymmetric selective transfer in multi-objective decision trees. In *Proceedings of the 11th International Conference on Discovery Science*, Budapest, Hungary, 2008.
17. Guo-Jun Qi, Xian-Sheng Hua, Yong Rui, Jinhui Tang, Tao Mei, and Hong-Jiang Zhang. Correlative multi-label video annotation. In *MULTIMEDIA '07: Proceedings of the 15th international conference on Multimedia*, pages 17–26, New York, NY, USA, 2007. ACM.
18. J. Read. A pruned problem transformation method for multi-label classification. In *Proc. 2008 New Zealand Computer Science Research Student Conference (NZCSRS 2008)*, pages 143–150, 2008.
19. Y. Schapire, R.E. Singer. Boostexter: a boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.
20. Cees G. M. Snoek, Marcel Worring, Jan C. van Gemert, Jan-Mark Geusebroek, and Arnold W. M. Smeulders. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, pages 421–430, New York, NY, USA, 2006. ACM.
21. Yang Song, Lu Zhang, and Lee C. Giles. A sparse gaussian processes classification framework for fast tag suggestions. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 93–102. ACM, 2008.
22. A. Srivastava and B. Zane-Ulman. Discovering recurring anomalies in text reports regarding complex space systems. In *IEEE Aerospace Conference*, 2005.
23. K.M. Ting and I.H. Witten. Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10:271–289, 1999.
24. K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas. Multilabel classification of music into emotions. In *Proc. 9th International Conference on Music Information Retrieval (ISMIR 2008), Philadelphia, PA, USA, 2008*, 2008.
25. G. Tsoumakas and I. Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13, 2007.
26. G. Tsoumakas, I. Katakis, and I. Vlahavas. A review of multi-label classification methods. In *Proceedings of the 2nd ADBIS Workshop on Data Mining and Knowledge Discovery (ADMKD 2006)*, pages 99–109, 2006.
27. G. Tsoumakas, I. Katakis, and I. Vlahavas. Effective and efficient multilabel classification in domains with large number of labels. In *Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08)*, pages 30–44, 2008.
28. G. Tsoumakas, I. Katakis, and I. Vlahavas. Mining multi-label data (accepted). In O. Maimon and L. Rokach, editors, *Data Mining and Knowledge Discovery Handbook*. Springer, 2nd edition, 2009.
29. G. Tsoumakas and I. Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. In *Proceedings of the 18th European Conference on Machine Learning (ECML 2007)*, pages 406–417, Warsaw, Poland, September 17-21 2007.
30. Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

31. David H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.
32. Ying Yang, G. I. Webb, J. Cerquides, K. B. Korb, J. Boughton, and Kai M. Ting. To select or to weigh: A comparative study of linear combination schemes for superparent-one-dependence estimators. *IEEE Transactions on Knowledge and Data Engineering*, 19(12):1652–1665, 2007.
33. Z. Younes, F. Aballah, and T. Denoeux. Multi-label classification algorithm derived from k-nearest neighbor rule with label dependencies. In *roceedings of the 16th European Signal Processing Conference*, August 2008.
34. M-L Zhang and Z-H Zhou. Multi-label neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1338–1351, 2006.
35. M-L Zhang and Z-H Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, 2007.
36. Yi Zhang, Samuel Burer, and W. Nick Street. Ensemble pruning via semi-definite programming. *Journal of Machine Learning Research*, 7:1315–1338, 2006.

Multi-label Classification by Analyzing Labels Dependencies

Lena Tenenboim, Lior Rokach, and Bracha Shapira

Department of Information Systems Engineering, Ben-Gurion University of the Negev
Deutsche Telekom Laboratories at Ben-Gurion University of the Negev,
P.O.B. 653, Beer-Sheva 84105, Israel
{lenat,liorrk,bshapira}@bgu.ac.il

Abstract. This research study deals with the improvement of multi-label classification by modeling existing dependencies between labels. The main purpose of the study is to define and develop a classification algorithm for multi-label classification tasks by partitioning the class set into several subsets. According to this algorithm, first, dependencies among the labels are analyzed and then the whole set of labels is divided into several mutually exclusive subsets. Finally, a classification algorithm incorporating dependencies among labels within each subset can be applied. An experimental study shows that the proposed method has high potential to achieve the defined objectives and improve multi-label classification performance.

1 Introduction

Regular classification tasks deals with classifying instances to a single label. In multi-label classification, the instances can be associated with a set of labels. Tsoumakas and Katakis [1] provide an exhaustive overview of the existing approaches for multi-label classification. They partition the existing methods for multi-label classification into two main categories:

1. Problem Transformation - methods that transform the multi-label classification problem into either one or more single-label classification or regression problems.
2. Algorithm Adaptation - methods that extend specific learning algorithms in order to handle multi-label data directly. Examples of the application of Algorithm Adaptation methods for multi-label problems in the literature can be found in [2–11]. The main disadvantage of Algorithm Adaptation methods is that their application requires changes of known classification algorithms in order to adapt them to a specific problem.

As for Problem Transformation, the common methods used are the Binary and Labels Power-set (LP) approaches. According to the LP approach each different set of labels that exists in the multi-label dataset is considered as a single label. The main disadvantage of this method is data sparseness: datasets created based on this approach are likely to have a large number of classes and only a few examples per class.

This issue was recently addressed in [12]. The authors propose Pruned Sets (PS) and Ensembles of Pruned Sets (EPS) methods to concentrate on only the most important correlations. This is achieved by pruning away infrequently occurring label sets; instances having a label combination occurring fewer times than the pruning parameter are rejected. The rejected examples are then post processed and partially reintroduced into the data by decomposing them into more frequently occurring label subsets. Finally a process similar to the regular LP approach is applied on the new dataset. This set is supposed to contain a limited number of frequent combinations. The authors show empirically that the proposed methods are often superior to other multi-label methods over a range of multi-labelled datasets. However, they are likely to be inefficient in domains with large Proportion of Distinct label combinations [12] and an even distribution of examples over those combinations. Another limitation of the PS and EPS methods is the need to balance the trade-off between information loss and adding too many examples with smaller label sets. Also, the correlations within the decomposed label sets are not considered.

According to the Binary approach, a multi-labeled classification problem is decomposed into multiple, independent binary classification problems, and the final labels for each data point are determined by aggregating the classification results from all binary classifiers. The main problem of this method is that possible dependencies among the labels are ignored. Also, when a large number of binary classifiers have to be learned, memory and classification time problems can result.

This issue was recently addressed in [13] where a new approach dealing with multi-label classification in domains with a large number of labels was proposed. The proposed algorithm (HOMER) organizes all labels into a tree-shaped hierarchy with a much smaller set of labels at each node. Each non-leaf node contains a disjoint subset of initial labels which is labeled with a new meta-label. A multi-label classifier is then constructed at each non-leaf node, following the Binary approach. The multi-label classification is performed recursively in a top-down manner. It starts from the root by classification into one or more meta-labels and proceeds into the child nodes only if their labels are among those predicted by the parent's classifier. One of the main HOMER processes is the clustering of the label set into disjoint subsets so that similar labels are placed together. This is accomplished by applying a *balanced k means* clustering algorithm on the label part of the data. Empirical evaluation showed that HOMER provides more accurate and quicker predictions than the popular Binary approach. However, it also ignores possible dependencies among the labels within each node.

An idea relatively close to that proposed in this research is presented in [14]. Tsoumakas and Vlahavas propose an approach that constructs an ensemble of LP classifiers. Each LP classifier is trained using a different small random subset of the set of labels. This approach (RAKEL) aims at taking into account label correlations and at the same time avoiding the problems of LP mentioned above. A comparison shows that the performance is better than that achieved by applying the popular Binary Relevance and LP methods on the full set of labels. Tsoumakas and Vlahavas note that the random nature of their method may lead to the inclusion of models that affect the ensemble's performance in a negative way.

This research aims to improve multi-label classification accuracy and computation cost by defining and developing classification methods that discover and consider possible dependencies among labels. We propose to discover existing dependencies among labels, in advance, before any classifiers are induced, and then to use the discovered dependencies to construct a multi-label classifier. This approach will overcome the main disadvantages of the common solutions, such as independence assumption and data sparseness. The proposed approach should improve the accuracy of multi-labeled classification, as it tries to find the optimal tradeoff between the simplicity of the Binary and the complexity of the Labels Power-set approaches.

In order to obtain an initial estimation of the validity of our hypotheses and a justification of our proposed methods, we ran some preliminary tests. During the tests we compared results (in terms of classification effectiveness) of common Binary and LP approaches with the results potentially achieved by the proposed methods. The results show that the proposed methods can achieve superior results compared to popular standard approaches.

The rest of the paper is structured as follows: the next section outlines the research objectives and hypothesis. We then describe the methods proposed for identification of dependencies among labels and multi-label classification incorporating the discovered dependencies. Finally, we describe the performed preliminary tests and present their results.

2 Research Objectives and Hypothesis

This section describes the objectives and the hypotheses of the research. Our hypotheses are based on the important concepts of *bias* and *variance*. Recall that *bias* measures the distance between the predictions of a learning algorithm for an example and the target value. *Variance* measures the effect of a training set on the predictions of the learning algorithm.

In general, bias and variance both depend on the complexity of the model but in opposite directions. Models that perfectly describe a specific learning set will not necessarily perform well on unseen data. As the fitness of a model to a learning set increases, the generality of the model decreases and the model is said to be "over-fitted" to the dataset. Thus, we assume that there exists an optimal tradeoff between these two sources of error.

2.1 Research Objectives

The main objective of this research is to improve multi-label classification accuracy and time performance by defining and developing classification methods that consider possible dependencies among labels and reduce the bias-variance error source.

We will define specific heuristic methods for all steps of the proposed classification process, namely: discovering dependencies among labels; clustering the labels into mutually exclusive subsets; and classification incorporating discovered dependent combinations. When defining the above methods we will ensure that the training and classification time will remain acceptable and as short as possible. Then, we plan to automate all the defined methods allowing their simple application to any classification problem, and to examine the effectiveness of the proposed methods by running classification experiments. Based on an analysis of the achieved results, we hope to be able to define some rules of thumb for identifying the best multi-label method in terms of the accuracy of results based on different context properties (e.g., number of concepts, training set size, label density and cardinality, application requirements, etc.).

2.2 Hypotheses

Referring to the research goals detailed above we define the following hypotheses:

Hypothesis 1: The proposed Dependent Clusters approach for multi-label classification performs better (in terms of accuracy) than Binary approach.

Hypothesis 2: The proposed Dependent Clusters approach for multi-label classification performs better (in terms of accuracy) than Labels Power-set approach.

Hypothesis 3: The proposed Dependent Clusters approach for multi-label classification results in a 'middle-complexity' model allowing better tradeoff between bias and variance in terms of accuracy measures.

The binary approach to multi-label classification results in a 'simple' model with high variance and bias. Ignoring the facts of category relations and treating single-labeled and multi-labeled items equally simplifies the model and generalize its results. Thus, the overall level of classification error of the process when applying this approach will be high due to many incorrect or partially correct results.

The Labels Power-set approach to multi-label classification results in a 'complicated' model with low bias and high variance. Modeling all possible combinations of labels will increase dependency of a model on specific samples presented in the learning set. Also, all eventual noisy relations will be modeled, and thus the overall classification error level will remain high.

The proposed Dependent Clusters approach for multi-label classification results in a 'middle-complexity' model allowing optimal tradeoff between bias and variance. Considering only a relevant part of all possible combinations of labels, i.e., only highly (or most) dependent labels, we generate a model that on the one hand treats the new multi-label examples in a sophisticated manner, and on the other is not "over-fitted" to the learning set. Thus, by achieving minimal bias and variance error contributors, the overall classification error level of the model will be reduced.

Hypothesis 4: In some specific cases of classification problems, the Binary and Label Power-set approaches can perform better than the proposed method. Among such specific cases are (1) small training set and (2) few labels with sufficient training examples for all existing combinations.

For a small training set it is important to produce a more general model, and thus in such a case the Binary approach will cope better with the problem. A reasonable number of label combinations and sufficient training examples for each combination eliminate the main problem of the LP approach and hint that all combinations are important and not incidental. Thus the LP approach is expected to produce better results than just partial modeling of dependencies and also is feasible in this case.

3 Research Methods

To achieve the research goals presented in the previous section we defined some heuristic methods based on statistical analysis of datasets. These methods allow rapid and simple modeling of category dependencies and are supposed to lead to a solution that is reasonably close to the best possible one.

The proposed approach consists of two main steps. The aim of the first step is to identify preliminary dependencies and to cluster all labels into several independent subsets. The second step is actually a multi-label classification incorporating the category dependencies discovered at the previous step. For each step we propose some possible methods. The proposed methods for the both steps are described below. In the framework of this research study we will implement and evaluate each one of the proposed methods and draw conclusions about its effectiveness depending on the applications domain and classification algorithms.

3.1 Dependencies Identification and Clustering

The first step towards multi-label classification incorporating dependencies among labels is dependencies identification and clustering of labels into independent groups. In this section we describe the methods proposed for this step.
Dependencies Identification. Our approach to multi-label classification is based on identifying related labels by analyzing data available in a training set before constructing the classifier. Once the dependent labels are identified we can consider the rest of the labels as independent according to the common binary approach. We plan to evaluate the following methods for identification of dependencies among labels:

1. Labels distribution analysis
2. Features among category distribution analysis
3. Category combinations shown in the training set
4. Supervised definition of dependencies.

Labels distribution analysis. The first proposed method is based on an analysis of the number of items in each category. This analysis can be carried out by applying one of statistical tests for independence (e.g., Chi-square, Fisher exact test, difference in proportions, likelihood-ratio-test) on the number of instances for each possible category combination. The analysis should begin by discovering related pairs of labels and can then continue to discover triples, quadruples, etc. of related labels. The analysis stops when no more dependencies can be discovered. Alternatively, we can define a maximum number of labels in a cluster and stop when this number is reached and all combinations of this number of labels are already analyzed.
Features among labels distribution analysis. This method is based on the analysis of features among items related to different labels. We can assume that classes are potentially dependent if they contain a certain number of common features. The number of common features depends on the total number of features in the model and should be defined empirically. Also, we will need to define what type of feature will be considered as relevant for the analysis, i.e., should we consider only representative features

having high weights in a class, or all features, regardless of their importance weight. For this analysis we can incorporate functions which measure the informative level of each feature. Such functions are also applied for dimensionality reduction purposes and are summarized in [15].

Labels combinations shown in training set. This method is based on the analysis of only those combinations of labels that are shown in the training set. To achieve this, any of the methods described above (i.e., category distribution analysis or feature distribution analysis) will be applied only to category combinations having at least one training example.

Supervised definition of dependencies. In some applications certain dependencies among labels can be known in advance. Such information could be used as a priori knowledge for modeling dependencies. Later, we can also use this knowledge to test the method proposed above.

Dependent labels clustering. When the dependent labels are identified, we need to cluster them into some groups, so all labels within the group are interdependent and labels in different groups are independent. Thus, each one of the groups will be treated as an independent set of labels. For this, an automatic clustering algorithm should be defined and implemented. We propose the following methods:

1. Execute some clustering algorithm that will analyze discovered dependencies and calculate the ”distance” function taking into consideration the class dependency strength (for example, alpha value if Chi-square test was applied) and the number of instances of co-occurrences in a certain class combination. Finally, the cluster with the best value of ”distance” function will be chosen. Thus we will find and cluster together the strongest dependency relations among all existing dependencies.
2. Use the Genetic Algorithm for exploring many grouping options and choose the one resulting in the best fitness function. For this we will need to define a genetic representation of the dependent class clusters and a fitness function, similarly to the “distance” function described above. This approach is expected to result in a combination modeling the strongest dependencies, as well.
3. Also, for clustering, a priori knowledge and a supervised definition can be available in some applications.

3.2 Multi-label classification incorporating labels dependencies

The second step is actually a multi-label classification incorporating category dependencies discovered at the previous step. In this section two methods proposed for carrying out this step are described. The first method is to apply a combination of standard Binary and Label Power-set approaches to the defined independent groups of labels. Another method is to train a meta-classifier that will predict additional labels for instances given a previously assigned label (or labels).

Binary and Label Power-set Combined Classification. One of the main disadvantages of the regular binary approach commonly applied to multi-label classification is that dependencies among labels are ignored. The main problem of the Label Power-set approach is data sparseness and the large number of label combinations. The proposed new approach applies combination of these common methods eliminating points where each of them suffers from its disadvantages.

Once we have identified the dependent labels and clustered them into independent groups, we can divide the classification responsibility in the following way:

- the Binary approach to the independent groups of labels will be applied without suffering from the problem of ignored dependencies;

- the Label Power-set approach to classification into labels within the dependent groups will be applied without incurring the problem of a large number of class combinations (as it is applied to a group with a limited, potentially small, number of classes).

For each independent group of labels, one single Label Power-set classifier is independently created and is used to determine to which of the labels from the group an instance belongs. In the case when a group includes only one category, the classifier will be binary. However, if a group consists of k labels, the classifier will, actually, be a single-label multi-class classifier with 2^k labels. Note that k is a number of maximum labels within one group and is in the control of the model designer. Eventually, the final classification prediction is determined, similarly to the Binary approach, by combining labels generated by each single Label Power-set classifier.

This approach: a) overpowers the independence assumption of the regular binary approach and b) allows simple multi-label classification using any readily available single-label classifier. The steps of the classification process according to the proposed approach will be as follows:

1. *Preprocessing and instances re-labeling*: All instances should be prepared for the classification according to the defined groups. For each independent group the full set of instances labeled according to category combinations within the group (following the LP approach) should be prepared.
2. *Classifier Training*: All the sets of instances prepared at the previous step will now be used to construct an independent single-label classifier for each independent group of labels. For this purpose any readily available classifier implementing some multi-class algorithm can be used.
3. *Classification*: each new instance will be processed by all independent classifiers and their single-label classification decision should be translated into names of initial labels.
4. *Results interpretation*: classification decisions received from each independent classifier are combined together in the final classification vector representing the multi-label classification result in terms of the initial set of labels.

Meta-classifier. Meta-learning methods, such as boosting and stacking, were previously successfully applied to multi-label classification problems [3] and [16]. These methods help to model dependencies implicitly existing in the dataset among labels and features. We propose applying the meta-learning techniques that explicitly provide dependencies among labels discovered earlier. The proposed approach is described below.

Generally, the meta-learning approach includes the construction of two types of classifier: a Base classifier, applied on row data; and a meta-classifier, a higher level classifier, applied to the results of base classifier.

We propose to use any readily available single-label classifier as our base classifier and then apply a number of meta-classifiers for multi-label classification on its result in the following way. First, a single-label classification of an instance is performed. The received classification decision is concatenated to the features' vector which is then processed by meta-classifiers to decide whether the instance, also, belongs to any other labels. The number of meta-classifiers that should be applied at each classification is equal to the number of independent groups of labels and denoted by m. According to this approach a group of m meta-classifiers for each category from the predefined set is required. Thus, the total number of meta-classifiers in the model will be $|L|$ * m. Although, the number of required models could be relatively high ($|L|^2$ in a worst case), the number of classifiers employed at each single classification is $1 + m$. As m can be of maximum of $|L|$, this simplifies to a worst case of $|L| + 1$.

This approach allows one to model dependencies among labels, provides multi-label classification using any readily available single-label classifier, and has a faster expected classification time than that of the Binary approach.

First, describe the proposed approach more precisely using a simple example. Then detail the required steps for the training and classification processes.

Let the whole set of labels L=$\{$A,B,C,D,E,F$\}$, |L|=6, and discovered groups of dependent labels be $\{$A,B,C$\}$, $\{$D,E $\}$, $\{$F$\}$. Thus, *m*, the number of independent groups and the number of meta-classifiers for each category, is 3. The total number of required meta-classifiers is 18 (=6*3). The constructed classifiers should be:

Base-level: any single-label classifier available to classify an instance x to one of $|L|$ labels.

Meta-level: for each category 3, single-label classifiers should be constructed, so that they can predict the relevance of other labels in all independent groups given a label result from the base classifier. The required classifiers are shown in Table 1, while notation h$\{$A,B|C$\}$ means a classifier available to predict the relevance of A and/or B labels, given label C.

Table 1. Example of required meta-classifiers.

Predefined label	Independent group		
	{A,B,C}	{D,E}	{F}
A	h{B,C|A}	h{D,E|A}	h{F|A}
B	h{A,C|B}	h{D,E|B}	h{F|B}
C	h{A,B|C}	h{D,E|C}	h{F|C}
D	h{A,B,C|D}	h{E|D}	h{F|D}
E	h{A,B,C|E}	h{D|E}	h{F|E}
F	h{A,B,C|F}	h{D,E|F}	-

Training process includes the construction of a base classifier and number of meta-classifiers.

Base classifier construction: a regular single-label classifier should be trained to predict one of labels in the predefined set.

Meta-classifiers construction: one meta-classifier should be constructed for each combination of a predefined category l and an independent group, except for a case when the predefined category is a single category in the independent group (see example in Table 1). For each such meta-classifier all instances should be pre-processed to prepare a special training set. In each training set, one feature column should be added to all instances and their label columns should be replaced by a new label. The value of the added feature corresponds to instance relevance to the related category l and is the same in all training sets for this category. The new label is created according to the instance relevance to the rest (except for l) of the categories in the independent group.

As an example of this process, consider the above labels set L along with the discovered independent groups of labels and training data of Table 2. New training sets required to induce meta-classifiers for category A are presented in Table 3. Similarly, training sets should be created for the other 5 labels of L.

Table 2. Example of a multi-label training set.

Example	Label set
1	{A,E}
2	{B,C}
3	{B}
4	{C,F}

Table 3. Example of training sets required to induce meta-classifiers for category A.

Example	An added feature	New labels in training sets for each independent group		
		{A,B,C}	{D,E}	{F}
1	1	¬B¬C	¬DE	¬F
2	0	BC	¬D¬E	¬F
3	0	B¬C	¬D¬E	¬F
4	0	¬BC	¬D¬E	F

Classification process: The steps of the classification process according to the proposed approach will be as follows. First, the base single-label classifier is applied to the instance x. Then, the classification decision of the base classifier is concatenated to the x features' vector and is processed by relevant m meta-classifiers to decide whether x, also, belongs to any other labels. The classification decisions of meta-classifiers should be translated into the names of initial $|L|$ labels, according to how it was coded in the training phase.

Results interpretation: the classification decision received from the base classifier is combined with the classification decisions of all the applied meta-classifiers.

4 Results

In order to obtain an estimation of the validity of our hypotheses and a justification of our proposed methods we performed the following experiments. The purpose of the performed tests was to compare the results (in terms of classification effectiveness) of the common Binary and LP approaches to multi-label classification with results potentially achieved by the proposed methods.

We tested the response of classification evaluation measures for different dependency combinations as a function of the training set size. By 'Dependency combination' we mean any possible combination of groups of correlated labels supposed to be discovered as the strongest combination during the Dependencies identification step.

The test was performed using Weka [17] and Mulan [14] software and was applied on Scene [18] and Emotions [19] datasets. These datasets have six distinct labels; each instance can be classified to one or more of these labels. We used the C4.5 tree classifier provided by Weka as our underlying base classifier for binary and single-label classification. The tests were performed in 10 cross-validation format. For evaluation purposes we used multi-label measures, namely Accuracy [8], H-Loss [3] and Subset Accuracy [20]. Below we describe the test performed and analyze its results.

4.1 Test description

In order to test our hypotheses we wanted to:

1. Compare the performance of the Binary and LP approaches given different training set sizes.
2. Compare the performance of the LP approach (where full inter-dependency among labels is considered) with the performance of the proposed approach, where partial dependency among labels is considered, at different training set sizes.

To this end, we ran the test applying the proposed method, which combines the LP and Binary approaches, on all possible dependency combinations of the initial labels, including full independence (Binary approach) and full inter-dependency among labels (LP approach). This test was iterated k times, removing N random instances from the training set before each next iteration.

As there are six different labels, we obtained 203 possible dependency combinations, including Binary (six groups: each of one independent category) and LP (one group of six correlated labels). There are 2407 instances in the Scene dataset, and 593 in the Emotions dataset. Thus, for the Scene dataset we set the N random instances to be removed at each time at 200; and for the Emotions dataset at 100. In total we performed 12 iterations (from 2407 to 207 instances) for the Scene dataset and 5 iterations (from 593 to 193 instances) for the Emotions dataset over all 203 combinations.

4.2 Results Evaluation

Results (Accuracy, Subset Accuracy and H-Loss) of the performed tests for some arbitrary chosen combinations are presented in Figure 1. The Scene dataset is seen on the left and Emotions dataset on the right.

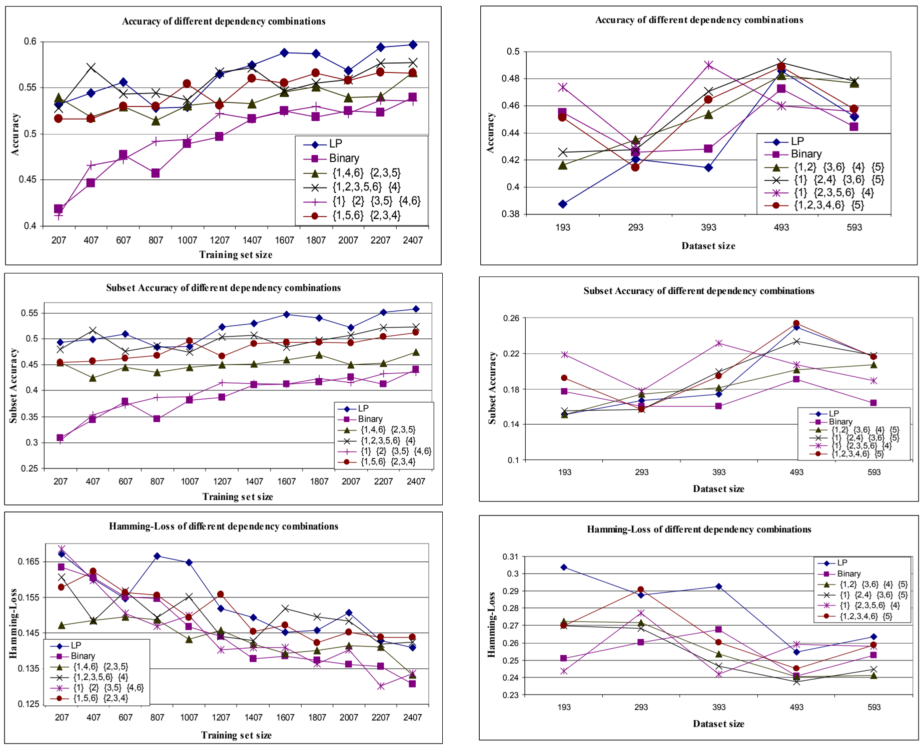


Fig. 1. Accuracy, Subset Accuracy and Hamming-Loss of different dependency combinations as a function of training set size (left- Scene dataset, right - Emotions dataset).

Firstly, consider the Scene dataset results. It can be seen from the graphs that the Accuracy and Subset Accuracy of LP are highest for large training sets and as the training set becomes smaller, other combinations provide similar or even better accuracy. The accuracy of BR is much worse for almost all combinations at all training set sizes, and it decreases much more drastically at smaller training sets. Considering the H-loss measure, providing an estimation of classification error, we can see that the error level of the LP approach is often relatively high compared to other combinations, especially when the size of the training set is reduced. So, also, the H-Loss of the Binary method has high values at small training set sizes. Other combinations of labels, especially the combination of two groups {1,4,6}, {2,3,5}, provide a stable level of error over different sizes of training set. Their H-Loss values are often lower than in the Binary and LP approaches.

Considering the Emotions dataset we ascribe it to medium or even small training sets as its full set contains only 593 instances. Even so we also evaluate it for some smaller training set sizes. From the graphs it can be seen that LP is outperformed by different combinations at almost all dataset sizes in terms of all three measures. The superiority of the different combinations over LP increases at smaller dataset sizes. The

Binary approach results in worse Accuracy and Subset Accuracy measures on the full dataset. However on smaller datasets it outperforms LP and some of other combinations in terms of all three measures.

The results for both datasets strongly support our hypotheses 1 and 2 that the proposed Dependent Clusters approach can provide results better that the Binary one and in many cases close to or even better than the LP approach. The results also support hypothesis 4(2) that the LP approach is preferable if a rich training set is available. Results for the Emotions dataset also support hypothesis 4(1) that the Binary approach could be preferable for small training sets. However, this hypothesis is not supported by the results for the Scene dataset. We plan to investigate this issue using additional datasets.

Surprisingly, the H-Loss of the Binary method in many cases on both datasets has values relatively low compared to those of LP and other combinations. This result contradicts our hypothesis 1 that the simplicity of the Binary model causes more errors. This issue should be further investigated using additional datasets.

We also present some graphs demonstrating the correlation between Accuracy and H-Loss measures for different training set sizes of the Scene dataset. The graphs for the full dataset with 2407 instances, for the medium dataset with 1207 instances and for the small dataset with 207 instances are presented in Figure 2. The points on the graphs mark the Accuracy and H-loss measures obtained for different group combinations.

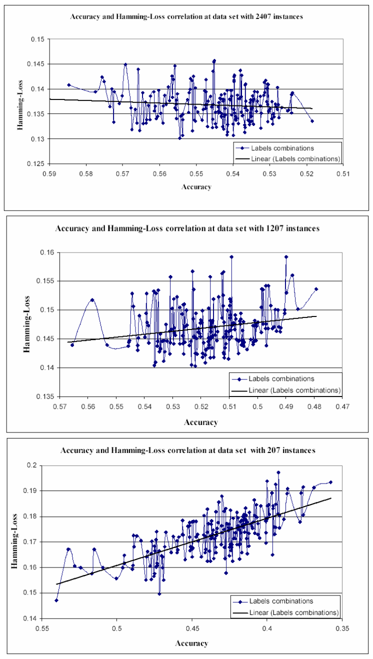


Fig. 2. The correlation between Accuracy and Hamming-Loss measures for the training sets with 2407, 1207 and 207 instances (Scene dataset).

We can see that for small and medium dataset sizes the general tendency of the correlatio n between Accuracy and H-Loss measures is as expected, i.e., for lower Accuracy values the H-Loss values are higher. However, for the full dataset size, the tendency of Accuracy and H-Loss measure is the opposite. There are higher values of H-Loss at higher Accuracy results. Also, we observe a strong correlation between the H-loss and Accuracy measure for small datasets and that this correlation reduces as the dataset size increases.

As for the Accuracy vs. H-Loss graphs of the Emotions dataset we will not present them here. Just let it be noted that the correlation between those measures is as expected (higher H-Loss values at lower Accuracy) for all training set sizes of this dataset. Recall that the Emotions dataset is considered to be of small to medium size.

From these observations we conclude that the correlation between Accuracy and Hamming-Loss is dataset-dependent, which is a point of interest for further investigation.

Consider the Accuracy and Hamming-Loss measures of the selected combinations at different training set sizes in both datasets (see graphs in Figure 3). From the graphs

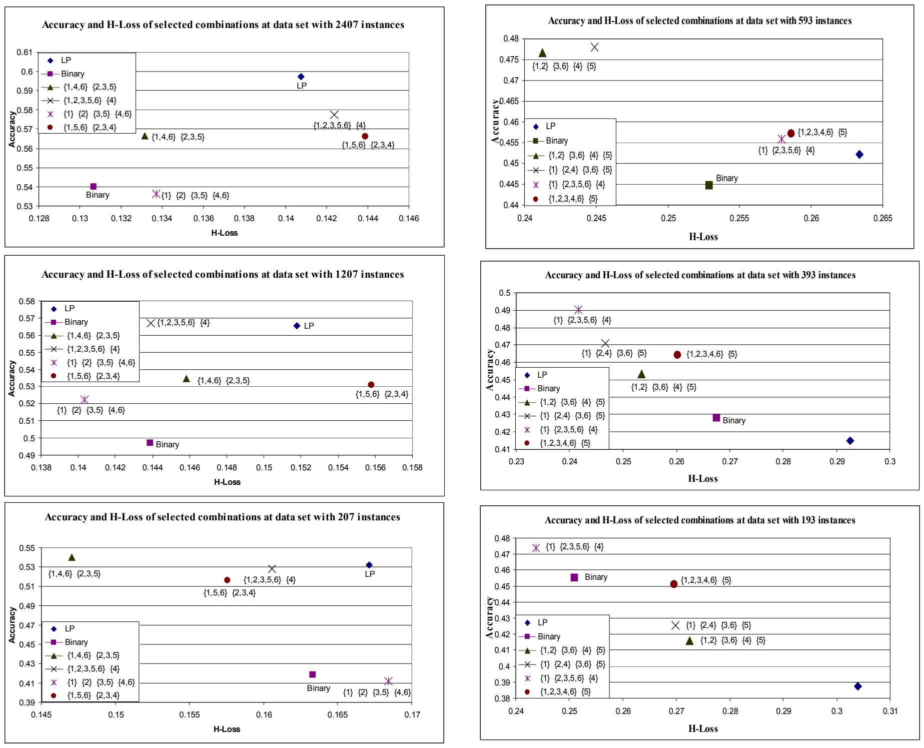


Fig. 3. Accuracy and H-Loss of selected combinations for Scene (left) and Emotions (right) training sets of different sizes.

we can see that (i) the Binary approach results in low Accuracy in almost all cases, except for the smallest training set of the Emotions dataset; (ii) the LP approach results in high H-Loss values for all training set sizes in both datasets and in low Accuracy values in all Emotions training sets; (iii) some combinations of independent groups result in high Accuracy and low H-Loss values for all training set sizes in both datasets. From these observations we conclude that the clustering of labels into independent groups of related categories may improve Accuracy and reduce H-Loss simultaneously, especially for small training set sizes.

These results strongly support our hypothesis 3 that the proposed Dependent Clusters approach results in a 'middle-complexity' model reducing overall classification error and improving Accuracy.

5 Conclusions

From all the above results, we can conclude that the proposed approach provides a classification accuracy that is competitive with or even better than the LP approach. Also, the proposed approach can be applied to problems where the LP approach is not feasible (when the labels set is too large) while providing an accuracy rate which is higher than that of the binary approach.

In this paper we examined the proposed method in two datasets. In order to validate the results, additional experiments with more datasets are required. Also, it will be interesting to compare our approach with some recent multi-label classification methods, such as RAKEL, EPS, and HOMER.

Moreover, other methods for identifying the label dependencies should be analyzed, and tested. Finally, other methods for classification incorporating the identified dependencies should be examined.

References

1. Tsoumakas, G., Katakis, I.: Multi-label classification: an overview. International Journal of Data Warehousing and Mining, 3(2007) 1–13
2. Clare, A., King, R.: Knowledge discovery in multi-label phenotype data. In: 5th European Conference on Principles of Data Mining and Knowledge Discovery. PKDD 2001, Freiburg, Germany (2001) 42–53
3. Schapire, R., Singer, Y.: BoosTexter: A Boosting-based System for Text Categorization. Machine Learning, 39(2/3) (2000) 135-168
4. Zhang, M.-L., Zhou, Z.-H.: A k-Nearest Neighbor Based Algorithm for Multi-label Classification. In: 1st IEEE International Conference on Granular Computing, 2(2005) 718-721
5. Luo, X., Zincir-Heywood, A.N.: Evaluation of Two Systems on Multi-class Multi-label Document Classification. In: 15th International Symposium on Methodologies for Intelligent Systems (2005) 161-169
6. Bacan, H., Pandzic, I.S., Gulija, D.: Automated News Item Categorization. In: 19th Annual Conference of The Japanese Society for Artificial Intelligence. (2005)
7. Elisseeff, A., Weston, J.: A kernel method for multi-labelled classification. In: Advances in Neural Information Processing Systems 14, MIT Press, (2001) 681-687

8. Godbole, S., Sarawagi, S.: Discriminative methods for multi-labeled classification. In: 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining, (2004) 22–30
9. Bergqvist, P., Eneroth, H., Hinz, J., Krevers, R., Laden, G., Mellbratt, A.: Categorization and Visualization of News Items. Cognitive Science Program, University of Linkoping, (2006)
10. Zhu, S., Ji, X., Xu, W., Gong, Y.: Multi-labelled Classification Using Maximum Entropy Method. In: 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM Press, (2005) 274-281
11. Vens, C., Struyf, J., Schietgat, L., Deroski, S., Blockeel, H.: Decision trees for hierarchical multi-label classification. Machine Learning, 73(2)(2008) 185–214
12. Read, J., Pfahringer, B., Holmes, G.: Multi-label classification using ensembles of pruned sets. In: Eighth IEEE International Conference on Data Mining, (0) (2008) 995-1000
13. Tsoumakas, G., Katakis, I., Vlahavas, I.: Effective and Efficient Multilabel Classification in Domains with Large Number of Labels. In Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08) (2008) 30-44
14. Tsoumakas, G., Vlahavas, I.: Random k-Labelsets: An Ensemble Method for Multilabel Classification. In: 18th European Conference on Machine Learning, Warsaw, Poland. (2007) 406-417
15. Sebastiani, F.: Machine Learning in Automated Text Categorization. ACM Computing Surveys, 34(1), (2002) 1-47
16. Arbel, R. and Rokach, L., Classifier evaluation under limited resources, Pattern Recognition Letters, 27(14) (2006) :1619–1631.
17. Weka - Data Mining with Open Source Machine Learning Software in Java, http://www.cs.waikato.ac.nz/ml/weka/
18. Boutell, M., Luo, J., Shen, X., Brown, C.: Learning multi-label scene classification. Pattern Recognition, 37(9), (2004) 1757-1771
19. Trohidis, K., Tsoumakas, G., Kalliris, G., Vlahavas, I.: Multilabel classification of music into emotions. In Proc. 9th International Conference on Music Information Retrieval (ISMIR 2008), Philadelphia, PA, USA, (2008)
20. Ghamrawi, N., McCallum, A.: Collective multi-label classification. In: 2005 ACM Conference on Information and Knowledge Management, Bremen, Germany, (2005) 195-200

Weighted True Path Rule: a multilabel hierarchical algorithm for gene function prediction

Giorgio Valentini and Matteo Re

DSI, Dipartimento di Scienze dell' Informazione,
Università degli Studi di Milano,
Via Comelico 39, 20135 Milano, Italia.
{valentini,re}@dsi.unimi.it

Abstract. The genome-wide hierarchical classification of gene functions, using biomolecular data from high-throughput biotechnologies, is one of the central topics in bioinformatics and functional genomics. In this paper we present a multilabel hierarchical algorithm inspired by the "true path rule" that governs both the Gene Ontology and the Functional Catalogue (FunCat). In particular we propose an enhanced version of the *True Path Rule* (TPR) algorithm, by which we can control the flow of information between the classifiers of the hierarchical ensemble, thus allowing to tune the precision/recall characteristics of the overall hierarchical classification system. Results with the model organism *S. cerevisiae* show that the proposed method significantly improves on the basic version of the TPR algorithm, as well as on the *Hierarchical Top-down* and *Flat* ensembles.

1 Introduction

Gene function prediction is a multiclass, multilabel classification problem characterized by hundreds or thousands of functional classes structured according to a predefined hierarchy (a directed acyclic graph for the Gene Ontology [1] or a tree forest for FunCat [2]). Functional classes are usually unbalanced (with positive examples usually less than negatives), with labels that can be uncertain and in many cases unknown or only partially known.

From a general standpoint several approaches have been proposed for multilabel classification, with applications ranging from protein function classification, to music categorization and semantic scene classification [3].

Different approaches to the hierarchical multilabel classification of gene function have been proposed [4, 5], but schematically we can individuate two main research lines: a) structured-output methods, based on the joint kernelization of both input variables and output labels using perceptron-like learning algorithms [6] or maximum-margin algorithms [7]; b) ensemble methods by which different classifiers are trained to learn each class, and then combined to take into account the hierarchical relationships between functional classes [8–10].

Along this second line of research, we propose a multilabel ensemble algorithm, specialized for tree-structured taxonomies, to predict the functional classes of genes.

Our proposed approach is directly inspired by the *true path rule* that governs the annotations of both GO and FunCat taxonomies [1]:

> "An annotation for a class in the hierarchy is automatically transferred to its ancestors, while genes unannotated for a class cannot be annotated for its descendants".

According to this rule the proposed ensemble method is characterized by a two-way asymmetric flow of information that traverses the graph-structured ensemble: positive predictions for a node influence in a recursive way its ancestors, while negative predictions influence its offsprings. The resulting ensemble embeds the functional relationships between functional classes that characterize the hierarchical taxonomy.

The proposed method predicts the annotations of genes at the level of the entire taxonomy or considering specific subsets of the hierarchical functional classes, and provides probabilistic and structured predictions of gene annotations. Moreover, by tuning a single global parameter, it allows to regulate the trade-off between precision and recall that characterizes gene function prediction problems. We apply the *True Path Rule (TPR)* hierarchical ensemble methods to the prediction of gene functions in yeast, using probabilistic SVMs as base learners [11], but the algorithm is general enough to be used with any probabilistic base learner and with other model organisms. Considering that data integration is crucial to improve prediction performances [12], TPR ensembles can be easily integrated with state-of-the-art biomolecular data integration methods [13], such as vector-space integration [14], kernel fusion [15] or ensembles of learning machines [16], without any modification of the algorithmic scheme.

This paper is organized as follows: in Sect. 2 the ensemble method inspired by the *true path rule* is presented. Sect. 3 summarizes the experimental set-up, while Sect. 4 show genome-wide gene function prediction results obtained with the model organism *S. cerevisiae* using the proposed method compared with hierarchical top-down and "flat" ensemble approaches. The conclusions and future developments end the paper.

2 Methods

2.1 Basic Definitions

Genome-wide gene function prediction can be modeled as a hierarchical, multiclass and multilabel classification problem. Indeed a gene/gene product x can be assigned to one or more functional classes of the set $\Omega = \{\omega_1, \omega_2, \ldots, \omega_m\}$. The assignments can be coded through a vector of multilabels $\mathbf{y} =< y_1, y_2, \ldots, y_m >\in \{0,1\}^m$, by which if x belongs to class ω_j, then $y_j = 1$, otherwise $y_j = 0$.

In both the *Gene Ontology (GO)* and *FunCat* taxonomies the functional classes are structured according to a hierarchy and can be represented by a directed graph, where nodes correspond to classes, and arcs to relationships between classes. Hence the node corresponding to the class ω_i can be simply denoted by i. We represent the set of children nodes of i by $\mathrm{child}(i)$, and the set of its parents by $\mathrm{par}(i)$. Moreover $y_{child(i)}$ denotes the labels of the children classes of node i and analogously $y_{par(i)}$ denotes the labels of the parent classes of i. Note that in FunCat only one parent is permitted, since the overall hierarchy is a tree forest, while in the GO, more parents are allowed, because the relationships are structured according to a directed acyclic graph. A classifier $D : X \rightarrow \{0, 1\}^m$ computes the multilabel associated to each example $x \in X$, and $d_i(x) \in \{0, 1\}$ is the label predicted by the classifier for class ω_i. For the sake of simplicity if there is no ambiguity we represent $d_i(x)$ simply by d_i.

2.2 An algorithm inspired by the "True Path Rule"

In both FunCat and GO ontologies, genes annotated to a specific functional class automatically belong to all its ancestors. Moreover, in FunCat, if a gene is not annotated to a given class, none of its offsprings can be annotated [1].

These basic rules constitute the so called "True Path Rule" that govern both GO and FunCat. Fig. 1 illustrates an example of the application of the true path rule.

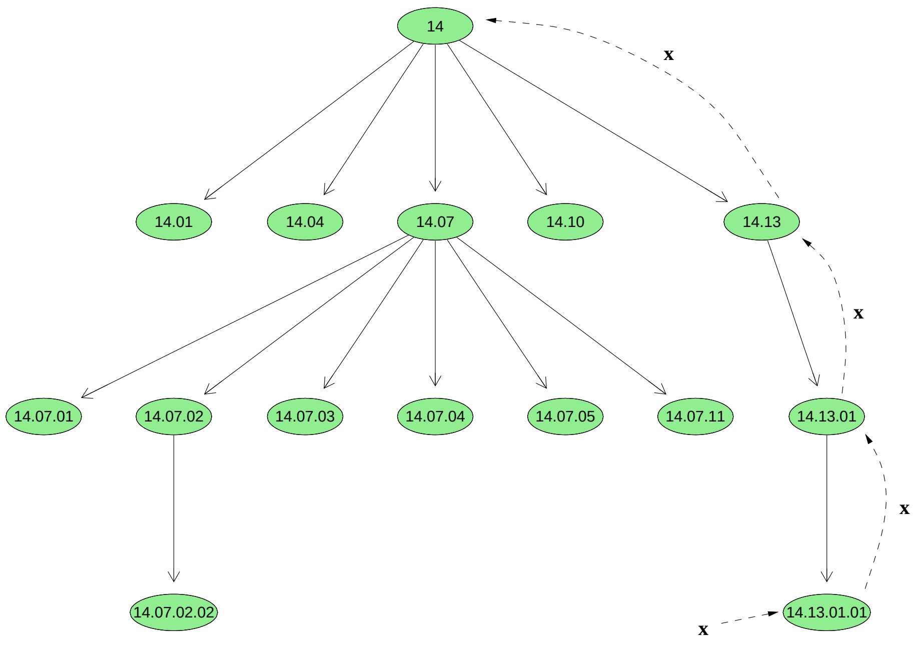


Fig. 1. FunCat tree rooted at class 14 (Protein fate): if example x belongs to class 14.03.01.01 then it belongs also to class 14.03.01, 14.03 and 14. On the contrary, if an example x does not belong to class 14.07 it cannot belong to any of its children (e.g. 14.07.01, 14.07.02, 14.07.03, 14.07.04, 14.07.05, 14.07.11).

[1] For the GO, this rule is slightly more complicated, because the GO is structured according to a directed acyclic graph, and even if a gene is not annotated to a class i, it can be annotated to a child of i, say j, if it is annotated to at least one of its parents $k \neq i$.

For a given example x, considering the parents of a given node i, a classifier that respects the true path rule needs to obey the following rules:

$$\begin{cases} d_i = 1 \Rightarrow d_{par(i)} = 1 \\ d_i = 0 \nRightarrow d_{par(i)} = 0 \end{cases} \tag{1}$$

On the other hand, considering the children of a given node i, a classifier that respects the true path rule needs to obey the following rules:

$$\begin{cases} d_i = 1 \nRightarrow d_{child(i)} = 1 \\ d_i = 0 \Rightarrow d_{child(i)} = 0 \end{cases} \tag{2}$$

The proposed hierarchical ensemble algorithm puts together the predictions made at each node by local ”base” classifiers to realize an ensemble that obeys the “true path rule”.

The basic ideas behind the *true path rule ensemble algorithm* can be summarized as follows:

1. Training of the base learners: for each node of the hierarchy a suitable learning algorithm (e.g. a multi-layer perceptron or a support vector machine) provides a classifier for the associated functional class
2. In the evaluation phase the trained classifiers associated to each class/node of the graph provide a local decision about the assignment of a given example to a given node.
3. Positive decisions may propagate from bottom to top across the graph: they influence the decisions of the parent nodes and of their ancestors in a recursive way, by traversing the graph towards higher level nodes/classes. On the contrary negative decisions do no affect decisions of the parent node (that is they do not propagate from bottom to top, eq. 1).
4. Negative predictions for a given node (taking into account the local decision of its descendants) are propagated to the descendants, to preserve the consistency of the hierarchy according to the true path rule. On the contrary positive decisions do not influence decisions of child nodes (eq. 2).

The ensemble combines the local predictions of the base learners associated to each node with the positive decisions that come from the bottom of the hierarchy, and with the negative decisions that spring from the higher level nodes. More precisely, base classifiers estimates local probabilities $\hat{p}_i(x)$ that a given example x belongs to class ω_i, but the core of the algorithm is represented by the evaluation phase, where the ensemble provides an estimate of the “consensus” global probability $p_i(x)$.

In [17] we proposed a basic algorithm based on the “True Path Rule” (the TPR algorithm), by which, given the set $\phi_i(x)$ of the children of node i for which we have a positive prediction for a given example x:

$$\phi_i(x) = \{j | j \in \text{child}(i), d_j(x) = 1\} \tag{3}$$

we can compute the consensus probability of the ensemble. The global consensus probability $p_i(x)$ of the ensemble depends both on the local prediction $\hat{p}_i(x)$ and on the prediction of the nodes belonging to $\phi_i(x)$:

$$p_i(x) = \frac{1}{1+|\phi_i(x)|}\left(\hat{p}_i(x) + \sum_{j\in\phi_i(x)} p_j(x)\right) \tag{4}$$

The decision $d_i(x)$ at node/class i is set to 1 if $p_i(x) > t$, and to 0 otherwise (a natural choice for t is 0.5). In the leaf nodes the sum of eq. 4 disappears and eq. 4 reduces to $p_i(x) = \hat{p}_i(x)$. In this way positive predictions propagate from bottom to top. On the contrary if for a given node $d_i(x) = 0$, then this decision is propagated to its subtree.

Note that with this basic version of the TPR algorithm there is no way to explicitly balance the local prediction $\hat{p}_i(x)$ at node i with the positive predictions coming from its offsprings (eq. 4). By balancing the local predictions with the positive predictions coming from the ensemble we can explicitly modulate the interplay between local and descendant predictors. To this end we introduce a *parent weight* w_p, $0 \leq w_p \leq 1$, such that if $w_p = 1$ the decision at node i depends only by the local predictor, otherwise the prediction is shared proportionally to w_p and $1 - w_p$ between respectively the local parent predictor and the set of its children:

$$p_i(x) = w_p \cdot \hat{p}_i(x) + \frac{1-w_p}{|\phi_i(x)|}\sum_{j\in\phi_i(x)} p_j(x) \tag{5}$$

In this way we can balance the weight of the prediction between the local component at node i and the component coming from its children, thus obtaining the *weighted TPR (TPR-w)* hierarchical ensemble algorithm.

The pseudocode of the TPR-w method is presented in Algorithm 1.

The algorithm is characterized by two main for loops: the external for (from row 1 to 30) handles a per level bottom-up traversal of the tree, while the internal (from row 2 to 29) scans the nodes at each level. If a node is a leaf (row 3), then the consensus probability p_i is equal to the local probability $\hat{p}_i(x)$. Note that a positive decision is taken if $p_i(x)$ is larger than a threshold t (row 5). If a node is not a leaf (row 10), at first the set $\phi_i(x)$ collects all the children nodes for which we have a positive prediction, and the consensus probability p_i of the ensemble is computed by considering the weighted local estimate of the probability $\hat{p}_i$ and the weighted probabilities computed by the children nodes for which a positive decision has been taken (row 13). In case of a negative decision for a node i, all the predictions relative to the subtree rooted at i are set to negative and their probabilities are set to p_i if larger than p_i. (rows 19-27). The algorithm provides both the multilabels associated to the example x and the probabilities p_i that a given example belongs to the class i, $1 \leq i \leq m$.

The bottom-up per level traversal of the tree assures that all the offsprings of a given node i are taken into account for the ensemble prediction. For the same reason we can safely set the classes belonging to the subtree rooted at i to negative, when $d_i(x)$ is set to 0. It is worth noting that we have a two-way asymmetric flow of information across the tree: positive predictions for a node influence its ancestors, while negative predictions influence its offsprings.

Algorithm 1 Weighted True Path Rule (TPR-w) hierarchical ensemble

Input:
- a test example x
- tree T of the m hierarchical classes
- set of m classifiers (one for each node) each predicting $\hat{p}_i(x),\ 1 \leq i \leq m$
- the weight w_p of the local prediction.

```
 1: for all levels $k$ of $T$ from bottom to top do
 2:     for all nodes $i$ at level $k$ do
 3:         if $i$ is a leaf then
 4:             $p_i(x) \leftarrow \hat{p}_i(x)$
 5:             if $p_i(x) > t$ then
 6:                 $d_i(x) \leftarrow 1$
 7:             else
 8:                 $d_i(x) \leftarrow 0$
 9:             end if
10:         else
11:             $\phi(x) \leftarrow \{j | j \in \text{child}(i), d_j(x) = 1\}$
12:             if $|\phi_i(x)| > 0$ then
13:                 $p_i(x) \leftarrow w_p \cdot \hat{p}_i(x) + \frac{1-w_p}{|\phi_i(x)|} \sum_{j \in \phi_i(x)} p_j(x)$
14:             else
15:                 $p_i(x) \leftarrow \hat{p}_i(x)$
16:             end if
17:             if $p_i(x) > t$ then
18:                 $d_i(x) \leftarrow 1$
19:             else
20:                 $d_i(x) \leftarrow 0$
21:                 for all $j \in$ subtree($i$) do
22:                     $d_j(x) \leftarrow 0$
23:                     if $p_j(x) > p_i(x)$ then
24:                         $p_j(x) \leftarrow p_i(x)$
25:                     end if
26:                 end for
27:             end if
28:         end if
29:     end for
30: end for
```

Output:
For each node i:
- the ensemble decisions $d_i(x) = \begin{cases} 1 & \text{if } x \text{ belongs to node } i \\ 0 & \text{otherwise} \end{cases}$
- the probabilities $p_i(x)$ that x belongs to the node $i \in T$

3 Experimental set-up

We predicted the functions of genes of the unicellular eukaryote *S. cerevisiae* using 7 different data sets and the *FunCat* taxonomy.

For each data set we evaluated the performance of four different ensembles: the *Flat* ensemble, that does not take into account the hierarchical structure of the data, the Hierarchical *Top-down* [18, 19], the basic True Path Rule (*TPR*) hierarchical ensemble and the proposed weighted variant (*TPR-w* described in the previous section). The hierarchical Top-down algorithm classifies an example x, where $d_i(x)$ is the classifier decision at node i and $root(T)$ denotes the set of nodes at the first level of the tree T, in the following way:

$$y_i = \begin{cases} d_i(x) \text{ if } i \in root(T) \\ d_i(x) \text{ if } i \notin root(T) \wedge y_{par(i)} = 1 \\ 0 \qquad \text{if } i \notin root(T) \wedge y_{par(i)} = 0 \end{cases}$$

For each ensemble we used as base learners linear Support Vector Machines (SVMs) with probabilistic output [11]. The performance of the ensembles have been compared using 5-fold cross-validation techniques. The selection of the w_p parameter in *TPR-w* ensembles have been performed by internal cross-validation. The threshold t of *TPR* ensembles has been set to 0.5 in all the experiments.

For the prediction of gene function in the yeast we used 7 bio-molecular data sets. For each data set we selected only the genes annotated to FunCat [2], using the *HCgene* R package [20]. We also removed the genes annotated only with the ”99” FunCat class (”Unclassified proteins”) and selected classes with at least 20 positive examples, in order to get a not too small set of positive examples for training. As negative examples we selected at each node/class genes not annotated to that node, but annotated to its parent. From the data sets we removed also uninformative features (e.g. features with the same value for all the available examples). At the end of these pre-processing steps we obtained data whose characteristics are summarized in Tab. 1.

Table 1. Data sets

Data set	Description	n.samples	n. feat.	n.class
Pfam-1	protein domain binary data from *Pfam* [21]	3529	4950	211
Pfam-2	protein domain log E data from *Pfam* [22]	3529	5724	211
Phylo	phylogenetic data [14]	2445	24	187
Expr	gene expression data [23, 24]	4532	250	230
PPI-BG	PPI data from *BioGRID* [25]	4531	5367	232
PPI-VM	PPI data from von Mering experiments [26]	2338	2559	177
SP-sim	Sequence pairwise similarity data [15]	3527	6349	211

Considering the unbalance between positive and negative examples, we adopted the classical F-score to jointly take into account the precision and recall of the ensemble for each class of the hierarchy.

[2] We used funcat-2.1 scheme, and funcat-2.1_data_20070316, available from the MIPS web site (http://mips.gsf.de/projects/funcat).

Moreover, we used also the *Hierarchical F-measure* that represents a generalization of the classical F-measure, in order to take into account the hierarchical nature of functional annotation [27].

Viewing a multilabel as a set of paths, hierarchical precision measures the average fraction of each predicted path that is covered by some true path for that gene. Conversely, hierarchical recall measures the average fraction of each true path that is covered by some predicted path for that gene. More precisely, given a general taxonomy G representing the graph of the functional classes, for a given gene/gene product x consider the graph $P(x) \subset G$ of the predicted classes and the graph $C(x)$ of the correct classes associated to x, and let be $l(P)$ the set of the leaves (nodes without children) of the graph P. Given a leaf $p \in P(x)$, let be $\uparrow p$ the set of ancestors of the node p that belong to $P(x)$, and given a leaf $c \in C(x)$, let be $\uparrow c$ the set of ancestors of the node c that belong to $C(x)$. The original definitions of Hierarchical Precision (HP), Hierarchical Recall (HR) and Hierarchical F-score (HF) [27], with the tree forests of FunCat can be simplified as follows:

$$\begin{aligned} HP &= \frac{1}{|l(P(x))|} \sum_{p \in l(P(x))} \frac{|C(x) \cap \uparrow p|}{|\uparrow p|} \\ HR &= \frac{1}{|l(C(x))|} \sum_{c \in l(C(x))} \frac{|\uparrow c \, \cap P(x)|}{|\uparrow c|} \\ HF &= \frac{2 \cdot HP \cdot HR}{HP + HR} \end{aligned} \tag{6}$$

An overall high hierarchical precision is indicative of most predictions being ancestors of the correct predictions, or in other words that the predictor is able to detect the most general functions of genes/gene products. On the other hand a high average hierarchical recall indicates that most predictions are successors of the actual, or that the predictors are able to detect the most specific functions of the genes.

4 Results

At first we compared the performance of ensemble methods considering the "per class" F-measure averaged across all FunCat classes for each data set. The results show that hierarchical methods largely outperform flat ensembles: flat ensembles obtain an average F-measure across the 7 data sets used in the experiments of 0.15 against respectively 0.22, 0.18 and 0.24 with Top-down, TPR and TPR-w ensembles (Tab. 2).

As explained in the experimental set-up (Sect. 3), the F hierarchical measure is a more appropriate performance metric for the hierarchical classification of gene functions. Tab. 3 shows that on the average TPR-w achieves the best results: 0.34 versus 0.25 (TPR) and 0.29 (Top-down ensembles). Note that TPR-w obtains equal or better results than Top-down ensembles with respect to all the data sets. More precisely considering 5-fold cross validation results for each of the 7 considered data sets TPR-w reported better results than Top-down at 0.05 significance level on 5 tasks, according to the 5-fold cross-validated paired t-test [28]. The basic TPR ensemble on the contrary

Table 2. Per class F-measure results. Flat: flat ensemble; HTD: Hierarchical Top-Down ensembles; TPR: True Path Rule hierarchical ensembles; TPR-w True Path Rule weighted hierarchical ensembles.

Data set	Flat	HTD	TPR	TPR-w
Pfam-1	0.2816	0.4041	0.3622	0.4037
Pfam-2	0.1153	0.2056	0.1562	0.2197
Phylo	0.0711	0.0067	0.0625	0.0906
Expr	0.0752	0.0623	0.0702	0.0773
PPI-BG	0.1730	0.2690	0.2344	0.2946
PPI-VM	0.2145	0.3589	0.2613	0.3558
SP-sim	0.1121	0.2489	0.1306	0.2540
Average	0.1489	0.2222	0.1824	0.2414

achieves slightly worse results than the Top-down. These results show that we need the weighted version of TPR ensembles to significantly enhance Top-down predictions.

Even if the main goal of this work consists in the development of a hierarchical algorithm that can be applied to the prediction of the overall taxonomy of a gene, we can restrict the analysis to specific subtrees of the taxonomy. For instance, Tab. 4 shows the results restricted to the subtree rooted at the “Metabolism” FunCat class (FunCat ID = 01, Fig.2).

A specific advantage of the TPR-w ensembles is the capability of tuning precision and recall rates, through the parameter parent-weight w_p (eq. 5). Fig. 3 shows, the hierarchical precision, recall and F-measure as functions of the parameter w_p. For small values of w_p (w_p can vary from 0 to 1) the weight of the decision of the parent local predictor is small, and the ensemble decision depends mainly by the positive predictions of the offsprings nodes (classifiers): as a consequence we obtain a higher hierarchical recall for the TPR-w ensemble. On the contrary higher values of w_p correspond to a higher weight of the “parent” local predictor, with a resulting higher precision. The

Table 3. Hierarchical F-measures results. HTD: Hierarchical Top-Down ensembles; TPR: True Path Rule hierarchical ensembles; TPR-w True Path Rule weighted hierarchical ensembles. Statistically significant difference at 0.05 significance level are in boldface.

Data set	HTD	TPR	TPR-w
Pfam-1	0.4123	0.3080	0.4131
Pfam-2	0.3406	0.2684	**0.3700**
Phylo	0.0497	0.2010	**0.2174**
Expr	0.1166	0.1696	**0.1784**
PPI-BG	0.3226	0.2670	**0.3485**
PPI-VM	0.3977	0.2796	0.4000
SP-sim	0.4251	0.2398	**0.4472**
Average	0.2949	0.2468	0.3392

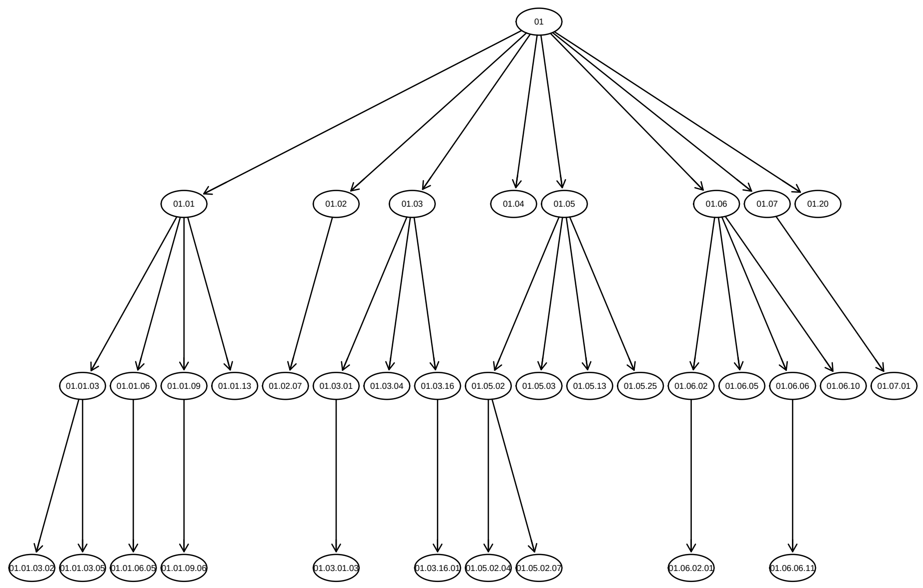


Fig. 2. Tree of the FunCat classes rooted at Funcat ID=01 (Metabolism).

opposite trends of precision and recall are quite clear in all graphs of Fig. 3. The best F-score is in "middle" values of the parameter parent-weight: in practice in most of the analyzed data sets the best F-measure is achieved for w_p between 0.5 and 0.8, but if we need higher recall rates (at the expense of the precision) we can choose lower w_p values, and higher values of w_p are needed if precision is our first aim. It is worth noting that we may vary the threshold t to obtain precision recall curves for a fixed value of w_p. In other words we may obtain different precision-recall curves for different value of w_p: the parent weight is a global parameter that affect the general precision/recall characteristics of the ensemble.

5 Conclusions

F hierarchical measures results show that TPR-w achieves equal or better results than the basic TPR algorithm and the Top-down hierarchical strategy, and all the hierarchical strategies achieve significantly better results than flat classification methods, using the classical "per-class" F-measure.

Another advantage of TPR-w consists in the possibility of tuning precision and recall by using a global strategy: large values of the parent weight improve the precision,

Table 4. Classification results on the FunCat tree rooted at "Metabolism", using Pfam-1 data. Each row represents a functional class of the FunCat taxonomy. Prec. stands for precision, Rec. recall, Sp. specificity, F F-measure, Acc. accuracy.

FunCat ID	Description	Prec.	Rec.	Sp.	F	Acc.
01	Metabolism	0.83	0.59	0.93	0.69	0.80
01.01	amino acid metabolism	0.62	0.34	0.98	0.45	0.94
01.01.03	assimilation of ammonia, metabolism of the glutamate group	0.27	0.15	0.99	0.19	0.98
01.01.03.02	metabolism of glutamate	0.37	0.32	0.99	0.34	0.99
01.01.03.05	metabolism of arginine	0.00	0.00	0.99	0.00	0.99
01.01.06	metabolism of the aspartate family	0.38	0.22	0.99	0.28	0.98
01.01.06.05	metabolism of methionine	0.53	0.29	0.99	0.37	0.99
01.01.09	metabolism of the cysteine - aromatic group	0.49	0.26	0.99	0.34	0.97
01.01.13	regulation of amino acid metabolism	0.10	0.03	0.99	0.05	0.98
01.02	nitrogen, sulfur and selenium metabolism	0.55	0.20	0.99	0.29	0.97
01.02.07	regulation of nitrogen, sulfur and selenium metabolism	0.27	0.11	0.99	0.16	0.99
01.03	nucleotide/nucleoside/nucleobase metabolism	0.65	0.35	0.98	0.46	0.95
01.03.01	purin nucleotide/nucleoside/nucleobase metabolism	0.72	0.40	0.99	0.52	0.98
01.03.01.03	purine nucleotide/nucleoside/nucleobase anabolism	0.61	0.29	0.99	0.39	0.99
01.03.04	pyrimidine nucleotide/nucleoside/nucleobase metabolism	0.63	0.42	0.99	0.51	0.98
01.03.16	polynucleotide degradation	0.52	0.27	0.99	0.36	0.98
01.03.16.01	RNA degradation	0.54	0.29	0.99	0.37	0.98
01.04	phosphate metabolism	0.81	0.61	0.98	0.70	0.94
01.05	C-compound and carbohydrate metabolism	0.79	0.50	0.97	0.61	0.91
01.05.02	sugar, glucoside, polyol and carboxylate metabolism	0.65	0.35	0.99	0.46	0.98
01.05.02.04	sugar, glucoside, polyol and carboxylate anabolism	0.55	0.33	0.99	0.41	0.99
01.05.02.07	sugar, glucoside, polyol and carboxylate catabolism	1.00	0.09	1.00	0.18	0.98
01.05.03	polysaccharide metabolism	0.78	0.25	0.99	0.38	0.98
01.05.25	regulation of C-compound and carbohydrate metabolism	0.47	0.16	0.99	0.24	0.96
01.06	lipid, fatty acid and isoprenoid metabolism	0.75	0.44	0.98	0.56	0.95
01.06.02	membrane lipid metabolism	0.76	0.41	0.99	0.54	0.98
01.06.02.01	phospholipid metabolism	0.69	0.36	0.99	0.48	0.98
01.06.05	fatty acid metabolism	0.42	0.15	0.99	0.22	0.99
01.06.06	isoprenoid metabolism	0.65	0.34	0.99	0.45	0.99
01.06.06.11	tetracyclic and pentacyclic triterpenes metabolism	0.61	0.23	0.99	0.34	0.99
01.06.10	regulation of lipid, fatty acid and isoprenoid metabolism	0.86	0.24	0.99	0.37	0.99
01.07	metabolism of vitamins, cofactors, and prosthetic groups	0.74	0.29	0.99	0.42	0.96
01.07.01	biosynthesis of vitamins, cofactors, and prosthetic groups	0.72	0.32	0.99	0.44	0.97
01.20	secondary metabolism	0.80	0.11	0.99	0.20	0.98

and small values the recall. The choice to favour precision or recall depends on the researcher's experimental objectives. In most data sets the best compromise between precision and recall is achieved for weights in the range between 0.5 and 0.8, that is giving a weight equal or larger to the local predictor with respect to the predictions taken by its offsprings.

Results show that also using a single source of evidence we can obtain a very high precision and recall for specific trees of the FunCat forest, but the overall results need to be improved for the genome-wide prediction of gene function. To this end, we need to integrate multiple data sources to obtain methods to predict function of hypothetical genes, or to discover or complete the functional annotation of genes whose function is incomplete or unknown. To this end the proposed approach can be easily integrated with at least three different general strategies for biomolecular data integration: vector space integration [14], kernel fusion [15] and ensemble methods [16]. Indeed for

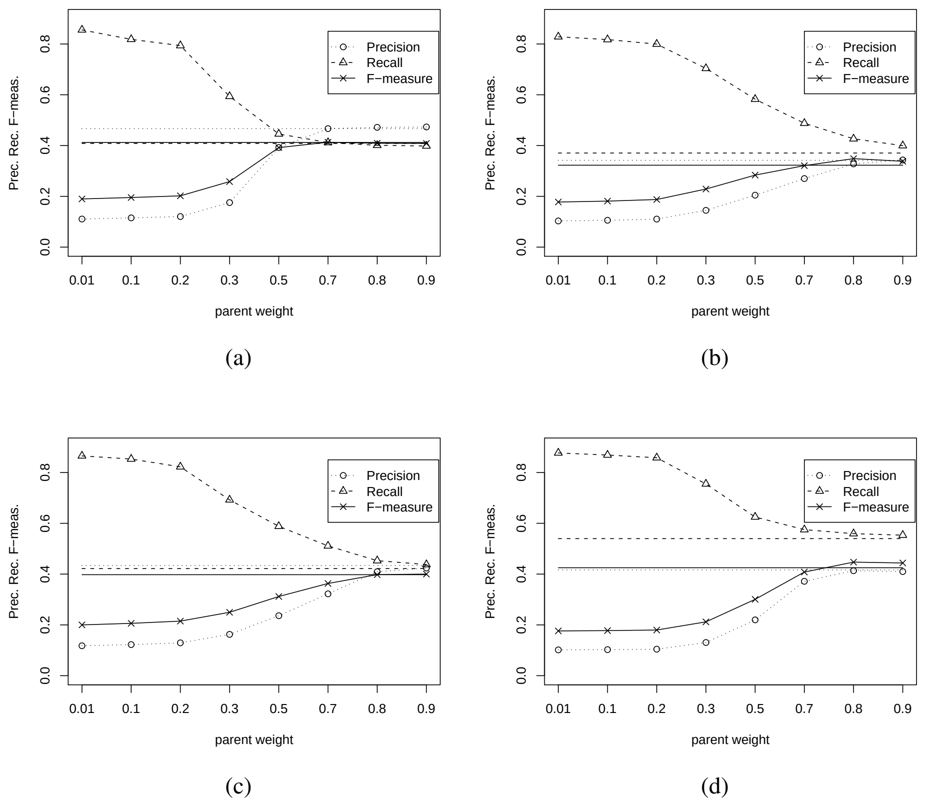


Fig. 3. Hierarchical Precision, Recall and F-measure as a function of the parent weight in TPR-w ensembles. Horizontal lines refers to top-down ensembles. (a) Protein domain binary data; (b) PPI BioGRID data; (c) PPI Von Mering data (d) Pairwise sequence similarity data.

each node/class of the tree we may substitute a classifier trained on a specific type of biomolecular data with a classifier trained on concatenated vectors of different data, or trained on a (weighted) sum of kernels, or with an ensemble of learners each trained on a different type of data.

Acknowledgments

The authors would like to thank the anonymous reviewers for their comments and suggestions, and gratefully acknowledge partial support by the PASCAL2 Network of Excellence under EC grant no. 216886. This publication only reflects the authors' views.

References

1. The Gene Ontology Consortium: Gene ontology: tool for the unification of biology. Nature Genet. **25** (2000) 25–29
2. Ruepp, A., Zollner, A., Maier, D., Albermann, K., Hani, J., Mokrejs, M., Tetko, I., Guldener, U., Mannhaupt, G., Munsterkotter, M., Mewes, H.: The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. Nucleic Acids Research **32** (2004) 5539–5545
3. Tsoumakas, G., Katakis, I.: Multi label classification: An overview. International Journal of Data Warehousing and Mining **3** (2007) 1–13
4. Barutcuoglu, Z., Schapire, R., Troyanskaya, O.: Hierarchical multi-label prediction of gene function. Bioinformatics **22** (2006) 830–836
5. Vens, C., Struyf, J., Schietgat, L., Dzeroski, S., Blockeel, H.: Decision trees for hierarchical multi-label classification. Machine Learning **73** (2008) 185–214
6. Sokolov, A., Ben-Hur, A.: A structured-outputs method for prediction of protein function. In: MLSB08, the Second International Workshop on Machine Learning in Systems Biology. (2008)
7. Astikainen, K., Holm, L., Pitkanen, E., Szedmak, S., Rousu, J.: Towards structured output prediction of enzyme function. BMC Proceedings **2** (2008)
8. Blockeel, H., Schietgat, L., Clare, A.: Hierarchcial multilabel classification trees for gene function prediction. In Rousu, J., Kaski, S., Ukkonen, E., eds.: Probabilistic Modeling and Machine Learning in Structural and Systems Biology, Tuusula, Finland, Helsinki University Printing House (2006)
9. Guan, Y., Myers, C., Hess, D., Barutcuoglu, Z., Caudy, A., Troyanskaya, O.: Predicting gene function in a hierarchical context with an ensemble of classifiers. Genome Biology **9** (2008)
10. Obozinski, G., Lanckriet, G., Grant, C., M., J., Noble, W.: Consistent probabilistic output for protein function prediction. Genome Biology **9** (2008)
11. Lin, H., Lin, C., Weng, R.: A note on Platt's probabilistic outputs for support vector machines. Machine Learning **68** (2007) 267–276
12. Valencia, A.: Automatic annotation of protein function. Curr. Opin. Struct. Biol. **15** (2005) 267–274
13. Noble, W., Ben-Hur, A.: Integrating information for protein function prediction. In Lengauer, T., ed.: Bioinformatics - From Genomes to Therapies. Volume 3. Wiley-VCH (2007) 1297–1314
14. Pavlidis, P., Weston, J., Cai, J., Noble, W.: Learning gene functional classification from multiple data. J. Comput. Biol. **9** (2002) 401–411
15. Lanckriet, G., De Bie, T., Cristianini, N., Jordan, M., Noble, W.: A statistical framework for genomic data fusion. Bioinformatics **20** (2004) 2626–2635
16. Re, M., Valentini, G.: Ensemble based data fusion for gene function prediction. In Kittler, J., Benediktsson, J., Roli, F., eds.: Multiple Classifier Systems. Eighth International Workshop, MCS 2009, Reykjavik, Iceland. Volume 5519 of Lecture Notes in Computer Science., Springer (2009) 448–457
17. Valentini, G.: True path rule hierarchical ensembles. In Kittler, J., Benediktsson, J., Roli, F., eds.: Multiple Classifier Systems. Eighth International Workshop, MCS 2009, Reykjavik, Iceland. Volume 5519 of Lecture Notes in Computer Science., Springer (2009) 232–241
18. Rousu, J., Saunders, C., Szdemak, S., Shawe-Taylor, J.: Learning hierarchical multi-category text classification models. In: Proc. of the 22nd Int. Conf. on Machine Learning, Omnipress (2005) 745–752
19. Cesa-Bianchi, N., Gentile, C., Tironi, A., Zaniboni, L.: Incremental algorithms for hierarchical classification. In: Advances in Neural Information Processing Systems. Volume 17., MIT Press (2005) 233–240

20. Valentini, G., Cesa-Bianchi, N.: Hcgene: a software tool to support the hierarchical classification of genes. Bioinformatics **24** (2008) 729–731
21. Deng, M., Chen, T., Sun, F.: An integrated probabilistic model for functional prediction of proteins. In: Proc 7th Int Conf Comp Mol Biol. (2003) 95–103
22. Finn, R., Tate, J., Mistry, J., Coggill, P., Sammut, J., Hotz, H., Ceric, G., Forslund, K., Eddy, S., Sonnhammer, E., Bateman, A.: The Pfam protein families database. Nucleic Acids Research **36** (2008) D281–D288
23. Spellman, P., et al.: Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomices cerevisiae by microarray hybridization. Mol. Biol. Cell **9** (1998) 3273–3297
24. Gasch, P., et al.: Genomic expression programs in the response of yeast cells to environmental changes. Mol.Biol.Cell **11** (2000) 4241–4257
25. Stark, C., Breitkreutz, B., Reguly, T., Boucher, L., Breitkreutz, A., Tyers, M.: BioGRID: a general repository for interaction datasets. Nucleic Acids Res. **34** (2006) D535–D539
26. von Mering, C., Krause, R., Snel, B., Cornell, M., Oliver, S., Fields, S., Bork, P.: Comparative assessment of large-scale data sets of protein-protein interactions. Nature **417** (2002) 399–403
27. Verspoor, K., Cohn, J., Mnizewski, S., Joslyn, C.: A categorization approach to automated ontological function annotation. Protein Science **15** (2006) 1544–1549
28. Dietterich, T.: Approximate statistical test for comparing supervised classification learning algorithms. Neural Computation **10** (1998) 1895–1924

Using Unsupervised Classifiers for Multilabel Classification in Open-Class-Set Scenarios[⋆]

Santiago D. Villalba and Pádraig Cunningham

School of Computer Science and Informatics, University College Dublin
{Santiago.Villalba,Padraig.Cunningham}@ucd.ie

Abstract. Multilabel classification is frequently reduced to binary classification via a simple one-versus-rest transformation. While supervised binary classification is the best understood and most successful strategy in machine learning, it is not exempt from practical problems that arise after this kind of transformation, for instance lack of robustness to class-skewness and noisy datasets or poor scalability. On the other hand, past research on multilabel classification has overlooked an issue that can be important on real applications: the existence of labels not represented in the training set. Therefore, methodological design for evaluation in this scenario is lacking. We present ongoing research on both aspects. We propose the application of one-class classifiers, binary unsupervised classifiers trained on positive examples only, as a theoretically appealing alternative, robust to both imbalanced datasets and the existence of unknown labels. The initial results are negative, showing the weakness of one-class classifiers. We also discuss possible pitfalls when evaluating classification performance on open-class-set problems.

1 Introduction

Amongst the approaches to solve the multilabel classification problem, a large body of the literature deals with reductions to standard binary classification (see [1] and references therein). A straightforward way of reorganizing the data to create binary classification problems is to follow a one-versus-rest approach, which is sometimes called Binary Relevance (BR) in multilabel domains. Following the notation in [1], let $L = \{\lambda_j : j = 1, \ldots, M\}$ be the finite set of labels in a multilabel learning task and $D = \{(x_i, Y_i) : i = 1, \ldots, N\}$ denote a set of inputs x_i and their corresponding outputs $Y_i \subseteq L$. For each λ_j, a binary classifier $c_j : X \to \{0, 1\}$ is trained using a binary dataset $D_j = \{(x_i, y_i) : (x_i, Y_i) \in D \wedge y_i = \mathbf{I}_{Y_i}(\lambda_j)\}$. Here $\mathbf{I}_A(x)$ is the indicator function, one if $x \in A$ and zero otherwise. At prediction time, label λ_j is included in the predicted set $\hat{Y}$ if c_j casts a positive prediction.

It is easy to prove that, for many of the usual multilabel classification performance criteria, we can achieve a minimum on the overall multilabel loss by just minimizing the loss on each individual label, and so the BR reduction is consistent for those losses. A reduction is said to be consistent if it yields an optimal predictor for the original problem when the auxiliary problems are also solved optimally [2]. In particular,

⋆ This research is supported by Enterprise Ireland Commercialisation Fund Grant No. CFTD/05/222.

label-based macro average measures, which include the Hamming-loss, render BR consistent. Those losses permit partial matches between predicted and actual label sets and are defined as the summation of the individual losses for each label; therefore the minimum is reached only when each different summand is minimum too. This contrasts with the analogous one-versus-rest reduction for crisp multiclass classification, that is not consistent as it requires classifier coordination; if a single base classifier casts a bad prediction, the whole multiclass classifier is deemed to fail unless a sophisticated decision function is applied.

Ultimately the appropriate loss depends on the application, but the consistency of BR for macro-averaged losses indicates that, in many cases, in order to maximize the performance of multilabel classifiers, one just needs to maximize the performance on each individual label. In those cases research in multilabel classification is just research into ways of getting closer to optimal binary classifiers. This is not to say that research in multilabel algorithms is futile, or that correlations between labels are not to be taken into account when training the classifier for each label. On the contrary, any single piece of information might be useful when training.

What is called for is to take into account the specific characteristics of the binary problems induced by the reduction from the multilabel setting, applying sensible corrections to any possible problems associated with this approach. Several deficiencies in the data, that can make supervised discriminative approaches fail, arise naturally from multilabel to binary reductions:

Imbalanced datasets. Because of the nature of the transformations and the typical high cardinality of the label set, usually the class distribution in the individual binary classification problems is highly skewed towards the negative class (where all the other labels have been put together). This creates known problems for the standard supervised classifiers [3] and so counter-measures (adjusting the loss function by adding costs, subsampling the majority class, oversampling the minority class) must be taken.

Noisy datasets. Directly relating to the quality of the data, if labeling is not consistent and complete for all the examples, contradictory goals can be given to the supervised crisp-classifier. In several of the most important domains where multi-label classification is necessary the process of generating labels can be highly subjective (for example emotion recognition or multimedia annotation) and so a gold standard is missing. If the base classifier is not robust to noise, performance can be severely damaged.

Scalability issues. In many cases computational costs become prohibitive in problems with a high number of labels.

Awareness of all the labels. Sometimes we do not have information about all possible labels (*i.e.*ifnextchar,latex@errorDont forget the comma!, an open-class-set scenario). This is important in domains where the label set is not static or there exists concept drift, like semantic multimedia annotation and network intrusion detection. A related question is whether our classifier must have an abstaining option. In multilabel classification this means equipping the classifier with the ability to predict the empty set. BR provides this option automatically, although sometimes it is regarded as bad behavior and therefore combatted[1] [4, 5].

If these are real problems in multilabel classification, what are suitable solutions? In section 2 we propose the use of one-class classifiers, that is, classifiers trained on positive examples only, as a possible approach to overcome some of the difficulties experienced by standard supervised classifiers after problem transformation. Then in section 3 we describe an exploratory experiment on open-class-set multilabel scenarios and study the applicability of one-class classifiers as building blocks in the BR algorithm, compared with standard supervised classifiers. We also discuss the methodological pitfalls that can arise in such experiments in section 4. The paper is brought to a conclusion in section 5.

2 One-Class and Multilabel Classifiers

One-class classifiers (OCC) [6] are classifiers trained with data from one class only, arbitrarily called positive. They are just normal binary classifiers that operate differently at training time by using only a subset of the data required to train conventional supervised classifiers. Therefore their application as building blocks for multilabel classifiers, for example as components for BR, is straightforward. The question is whether it is worthwhile.

One-class classification is motivated by a deficiency, not by the prospect of higher performance on conventional supervised problems. That deficiency can result from the quality of the raw material used to solve the classification problem: if the training data available is not representative of the concept to be learnt (the discrimination between classes) applying one-class classification is an option. That can be the case when the negatives space is too broad (*e.g.*ifnextchar,latex@errorDont forget the comma!, the writings of Cervantes against any other possible writing), when it is expensive to label the negatives (*e.g.*ifnextchar,latex@errorDont forget the comma!, multimedia annotation) or when negative examples have not yet arisen (*e.g.*ifnextchar,latex@errorDont forget the comma!, industrial process monitoring). In these cases building a discriminative model using the ill-defined negatives sample will lead to very poor generalization performance and therefore conventional supervised techniques are not appropriate (when usable).

Some other benefits, relevant to multilabel classification, can result from the application of one-class classifiers. As there is no need to trade-off between classes the decision boundaries are better specialized for each class and they do not suffer when classes are imbalanced in the training set. In fact, OCCs can be regarded as an extreme re-balancing process [7]. They naturally extend to open-class-set problems and despite suffering from normalization and combination issues, they are suitable as building blocks for multiclass classifiers with reject (predict “unseen class”) option [8].

They also scale gracefully with the number of labels and have zero-cost retraining when adding a new label in BR, as this only requires us to train a new classifier on the

[1] In fact, due to the class imbalance problem, often the individual binary classifiers become reject-all machines and so the prediction an the empty label set becomes a real problem.

data available for the new class. This contrasts with the cost of adding a new class in BR when using standard supervised classifiers. In this case we will need to retrain $|L|$ classifiers. If we assume for a moment that the average number of examples per class is $\bar{N} = N/|L|$ and that training a classifier is $O(f(N)) = O(f(|L|\bar{N}))$, then the whole operation is $O(|L|f(|L|\bar{N}))$. With the addition of new classes, supervised classifiers may not have the capacity to model the new space of concepts and so they must be adapted to the learning complexity of the problem. It appears to us that this is not such an important issue for one-class classifiers.

But one-class classification does not come at no cost and usually there is a high penalty in terms of generalization performance when compared with standard supervised classification on datasets where the motivational characteristics for one-class classification are not present. Usually one-class classifiers are much poorer than standard binary classifiers, as eliminating class dependencies from the training bias dismisses potentially critical information. One-class classification is, by definition, an ill-posed problem, as it tries to solve a discrimination problem without having direct information of what to discriminate against.

Operationally, one-class classification models are made up of two elements: the resemblance (or distance) function and the threshold. The first element to be considered is a measure for the distance $d(x)$ or the resemblance (probability) $r(x)$ of an example to the target class. The way those measures are computed vary from one approach to another, but finally we have a method for computing the degree of positiveness or negativeness ("outlierness") for any example. The second ingredient for one-class classification models is the threshold τ, a value that, when compared to the resemblance for a new example, determines whether it is classified as positive or not. Once we have trained the model to describe the positives, which includes the estimation of τ, the classification for an example is given by:

$$c(x) = \mathbf{I}(r(x) \geq \tau)$$

where $\mathbf{I}$ is an indicator function: 1 if x is to be accepted as positive and 0 otherwise. The most common approach is to define a *global* threshold, that is, a sole number that dictates the classification for a new example regardless of its nature or location. The standard way of computing this global threshold is by assuming that a certain (usually user supplied) percentage of the training data are outliers; in this way, it is enough to sort the values of the resemblance function for the training data and set the threshold to a value that will make the classifier reject the specified percentage of examples.

Threshold selection is directly related to the robustness and capital for one-class classifiers generalization capabilities. If it is too tight the number of false negatives will be increased; this can happen if the noise level specified by the user is too high. If it is too loose, the number of false positives will increase; this will happen if the noise level specified is too low. In either case one-class classifiers become reject-all or accept-all machines, which is a very common and undesirable effect.

This is not the first study that has considered the application of one-class classifiers in the multilabel context. An example is the work of Lee *et al*ifnextchar..[9]. They propose a multilabel system for solving a protein localization classification problem based on an extension to support-vector data description [6], a one-class classifier in the family of support vector machines. They claim the whole classification system to be well-adapted to multilabel problems where classes overlap [2] and report good performance, competitive with the state of the art.

In the next section we report an initial experiment on multilabel classification in an open-class-set scenario, assessing whether one-class classifiers used as components in a plain BR classifier provide any distinctive advantage, specially on handling the unknown classes.

3 An Experiment: Growing the Known Classes Set

In order to assess the performance of both supervised and one-class algorithms in a scenario where not all the classes are known, we perform a partition of the label set into two disjoint sets:

$$L = L_K \cup L_U, L_K \cap L_U = \emptyset$$

where L_K are the known labels and L_U are the unknown. We then split the known examples ($D_K = \{x_i \in D : Y_i \cap L_K \neq \emptyset\}$) into training and testing sets. We evaluate the performance using two different test sets: one containing known labels only, which is the usual setting, and a superset contaminated with unknown labels, violating the assumption that training and testing sets come from the same distribution. This methodology is applied in domains such as network intrusion detection, where classifiers should be able to detect new kinds of attacks [10].

3.1 Growing the Known Class Set

The number of possible label subsets is exponential in $|L|$. The empirical results can be highly dependent on label sampling artifacts for the known/unknown classes combinations, and it is difficult to ascertain whether a chosen set reflects the real characteristics of the domain. Different classifiers model better some classes than others and so the results are influenced by taking combinations of easy or difficult to separate classes from one set to another. In many cases there is also a high class skew with rare and common labels, and the size of the known labels set combined with the size of the classes it contains control the prior probability of known and unknown classes, which is an important parameter in assessing the trade-off between the performances in both samples.

A possible approach in practical research is to fix the proportion $|L_K|/|L_U|$ and randomize the choice of the known labels, running the evaluation several times. Here we follow a different, systematic approach when splitting the label set. We consider a total order of the labels in $L = \{\lambda_1, \ldots, \lambda_M\}$ and create a sequence of sets $L_K^{(1)} \subset L_K^{(2)} \subset \ldots \subset L_K^{(t)} = L$ by always adding only consecutive labels. For example, a possible sequence of known-label sets for $L = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$ is $L_K^{(1)} = \{\lambda_1, \lambda_2\}, L_K^{(2)} = Ł_K^{(1)} \cup \{\lambda_3\}, L_K^{(3)} = Ł_K^{(2)} \cup \{\lambda_4\}$.

This approach allows us to measure the performance over a range of known / unknown proportions. Running several repetitions with random label orderings would help

[2] The concept of class overlap is to some extent vague in a multilabel scenario.

to combat label-sampling biases, which we do not do in the following experiments due to computational time constraints. A fundamental question is whether this methodology is appropriate and to which extent it is free of artifacts, which is not clear to us yet and we discuss in section 4, after presenting some experiments.

3.2 Datasets and Preprocessing

In Table 1 we show a summary of the datasets[3] we use. The only preprocessing we perform is on the *bibtex* dataset, where each document is normalized to unit L_2 norm; even if the features in *bibtex* are just binary indicators of the presence of a word, this approach never harms and sometime substantially improves the performance of the classifiers.

Table 1. Summary of the datasets used in the evaluation. *Cardinality* is the average number of labels per example ($\frac{1}{N}\sum_{i=1}^{N}|Y_i|$), *Density* is the cardinality normalized by the number of labels ($\frac{1}{M}$*Cardinality*) and *Distinct* is the number of different label subsets present.

Dataset	Domain	Size	Dimensions	Labels	Cardinality	Density	Distinct	Source
emotions	music	593	72	6	1.25	0.311	27	[11]
scene	image	2407	294	6	1.07	0.179	15	[12]
yeast	biology	2417	103	14	4.24	0.303	198	[13]
bibtex	text	7395	1836	159	2.41	0.015	2856	[14]
mediamill	video	43907	120	101	4.38	0.043	6555	[15]

3.3 Classifiers

We compare the performance of six classifiers, three one-class classifiers and corresponding supervised models, as building blocks in the BR reduction.

Lazy learning (k-Nearest-Neighbours): The nearest neighbour approach can be used for constructing one-class classifiers. The training data is stored and an *outlierness* criterion is calculated for new examples based on their nearest neighbours, *i.e.*ifnextchar,latex@errorDont forget the comma!, their position relative to the seen examples. Several criteria have been proposed to measure the outlierness of an example. Here we use γ [16] which is the average of the distances to the k nearest neighbours (oc-kNN). We set the threshold via an inexpensive leave-one-out cross-validation of the scores in the training set combined with the usual fraction-of-rejection approach with a parameter of 0.01. The corresponding supervised technique is BRkNN (kNN) [5]. We set $k = 5$.

Generative model (Mixture of Gaussians): A simple mixture of 3 Gaussians with diagonal covariances (oc-MoG) is fitted to the positives and the emitted probability is used as the resemblance criterion. The threshold is set using the fraction-of-rejection criterion with a parameter of 0.01. The corresponding supervised technique fits a second mixture to the negatives and uses the probability emitted as the

[3] Available for download at `http://mlkd.csd.auth.gr/multilabel.html`

score for the negative class (MoGDA, standing for Mixture of Gaussians Discriminant Analysis).

Boundary Method (One-class SVM): We use the one-class ν-SVM [17] method (oc-SVM), that computes hyper-surfaces enclosing (most of) the positive data. We set ν, the regularization parameter that controls how much we expect our training data to be contaminated with outliers, to 0.01. As is common practice in OCC we use the Gaussian kernel, initializing the width of the kernel to the average pairwise Euclidean distance in the training set for *emotions*, *scene*, *yeast* and *mediamill*, and to the inverse of the dimensionality for *bibtex*. The corresponding supervised technique is standard C-SVM, with the same kernel parameters and $C = 1$; we also calibrate the scores using Platt scaling [18].

3.4 Results

We perform a 5-fold cross validation on the "small datasets" (*emotions*, *scene* and *yeast*) and a 75% train / 25% test partition in the "big ones" (*mediamill* and *bibtex*). We grow the known class set at regular intervals and report the evolution of the macro-averaged AUC and F1. In Figure 2 we show the results on the *scene* dataset, that are consistent with those observed on the rest of the datasets (see Figures 3 and 4). Several general observations can be made:

– Including unknown classes in the testing set provokes worse results for all classifiers but does not change the relative order in the classifier ranking for any of the datasets. For that reason we only report results in all-labels for *emotions*, *yeast*, *bibtex* and *mediamill*.
– The one-class classifier approaches are usually not competitive against the supervised ones. Even if expected, it is disappointing that they are not able to provide a distinctive advantage in scenarios with many unknown labels. Anyway one-class classifiers are fairly insensitive to the presence of unknown classes, as witnessed by their AUC performance curves being almost flat.
– The threshold selection policies for the one-class classifiers are especially weak. This is exemplified in the relation between oc-MoG and MoGDA for the F1 measure. They both have exactly the same potential as classifiers, since their scores for the positive class are the same. The only difference between them is in the thresholding policy, and it is the case that MoGDA always beats the fixed threshold policy of oc-MoG. Using thresholding biases based on the known negatives seem therefore a more reasonable approach to improve the performance of one-class classifiers.
– C-SVM is the overall winner amongst the supervised approaches in terms of ranking capabilities. In agreement with the results reported in [10], where SVMs have the best behavior on both known and unknown classes, this is probably due to the generalization capabilities of the SVM rooted on the margin maximization bias. However only in *bibtex* does that difference translate to actual classification improvements. For the F1 measure the winner is kNN, indicating that effort must also be made on improving the thresholding methodology for supervised approaches.
– kNN has the worst performance for AUC with small number of known labels and is always beaten by oc-kNN. This is due to kNN being constrained to report always

one of the known classes - there will always be a nearest neighbor to one of the examples in training set, even if it is very far. This suggest that a combination of both approaches could increase performance in both areas.

To illustrate the scalability at training time of the supervised approaches as opposed to that of one-class classification, we show in Figure 1 the training times of the six classifiers on the *bibtex* dataset.

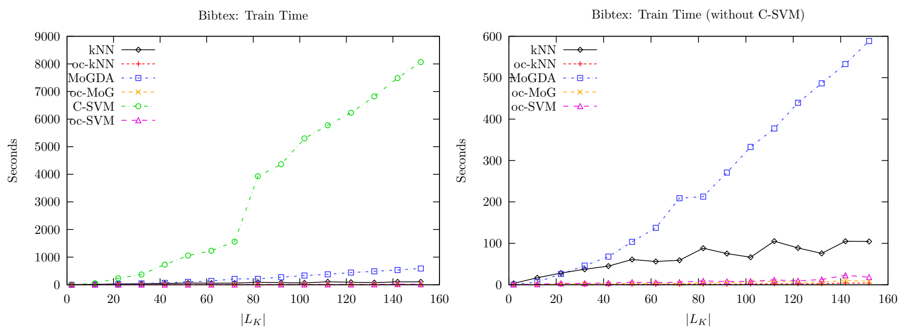


Fig. 1. Scalability of the classifiers on the *bibtex* dataset with increasing number of known labels. On the left, it is clear that the supervised SVM has the worst computational performance, as its training time is cubic on the size of the training set, and so it is orders of magnitude slower with a high number of labels. On the right, the second slower method is MoGDA. In any case, the one-class approaches have no problems on scaling in domains with many labels.

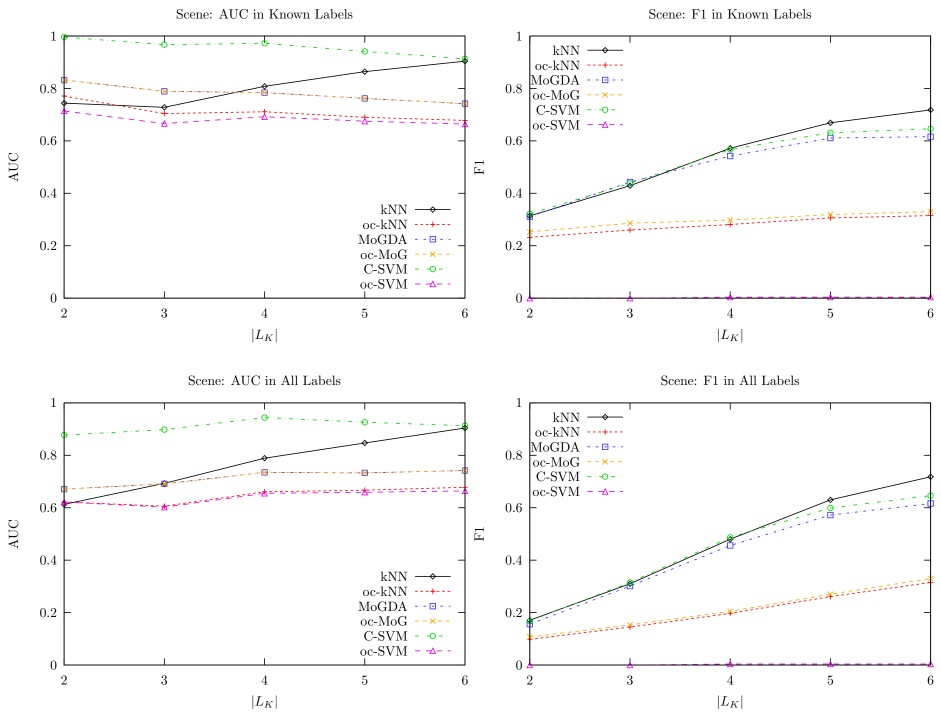


Fig. 2. Results for the *scene* dataset. On the top row is the test set with only known classes and on the bottom the test set augmented with examples from the unknown labels. On the left we report macro-AUC and on the right macro-F1. We appreciate that the performance on scenarios with unknown labels is much worse for all the classifiers, but that the relative performances between classifiers are constant. The relative positions of the curve with all the labels and the one with only known labels is consistent in all the analyzed datasets, so for the rest we only report the results on the test set with known and unknown labels.

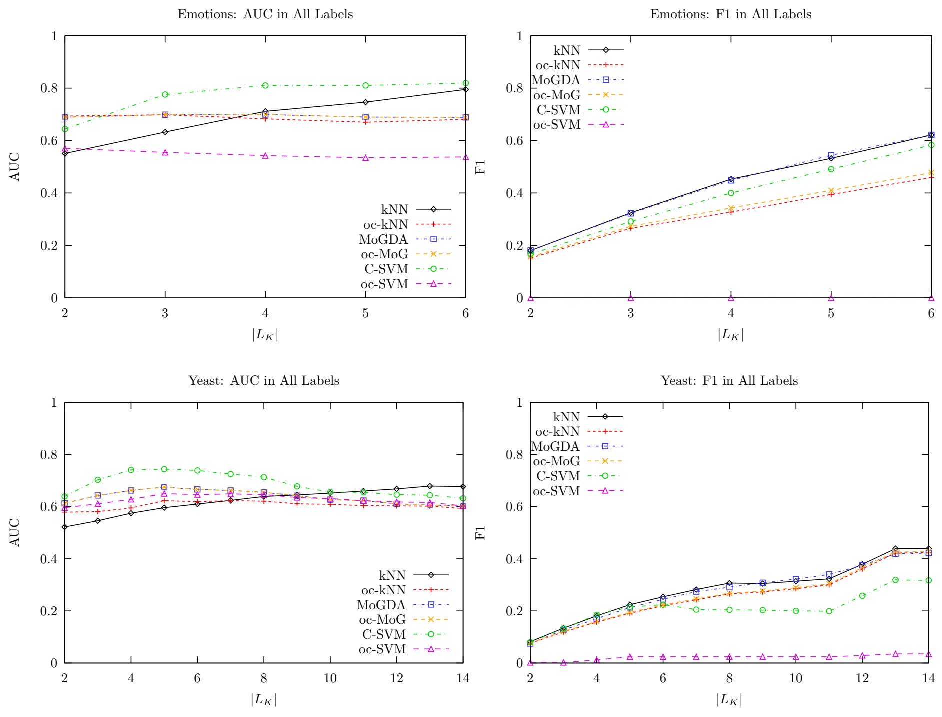


Fig. 3. Results for the *emotions* (top row) and *yeast* (bottom row) datasets. On the left we report macro-AUC and on the right macro-F1. The performance of oc-SVM is very poor, probably because of the lack of model selection and one of the key ingredients for the success of the supervised SVM: margin maximization.

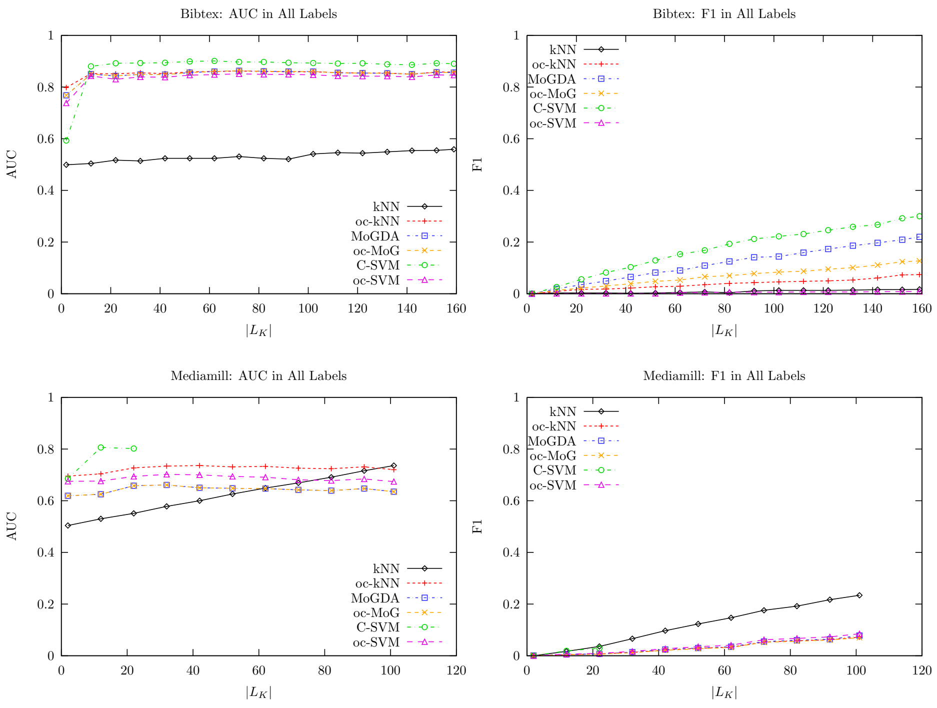


Fig. 4. Results for the *bibtex* (top row) and *mediamill* (bottom row) datasets. On the left we report macro-AUC and on the right macro-F1. For *mediamill* C-SVM is not able to complete in the time allotted even for a small subset of the labels, indicating its poor scalability. We regard the results of the one-class approaches on the *bibtex* dataset as promising.

4 More methodological issues

Adding an unknown class poses a serious challenge for performance evaluation. We cannot measure the performance on it alone, as the only defined measure is specificity (the true negative rate) because there are only negative instances of the unknown class. Measuring specificity alone is not an option, as classifiers that are reject-all machines will have a perfect score while more balanced models will probably perform worse. The option is to mix them with the known classes and measure a more balanced criterion in this test set. In our experiments we have observed that the specificity of supervised classifiers in the unknown set promptly becomes perfect, as they are biased towards predicting all negatives - the number of training negatives is soon overwhelming - while for one-class classifiers it depends on whether they are reject-all, accept-all or more balanced models. As specificity depends on a good thresholding policy, the option to measure the classifier scoring capabilities is to look at the differences in AUCs. In this case useful information on the performance can be hidden by the already present ranking of positive and negative examples, so an alternative is to measure AUC on positives and unknown negatives only.

There is an important question of whether we fix the test set size or not, and the composition of the test set can become both an artifact and an opportunity to gain insight into the tradeoffs between different proportions of the actual set of unknown negatives. We are not so comfortable with the idea of growing the known class set by taking classes from the unknown set, as that creates fluxes of examples from testing to training whose interaction with the performance measurement is not always neat. We cannot think of a sensible alternative if we want to study the problem under different proportions of known / unknown negatives other than fixing a set of unknown labels that are never included in training and grow the known set from the rest of the labels.

In Figure 5 we show two different ways of conducting the splitting for the methodology that we have followed in this paper. We call them “train vs test first” and “known versus unknown first”. The results reported in here followed the “train versus test first” splitting that dismisses the unknown examples in the train set. The relation between using this approach and that of always using all the possible unknown examples in the testing set are always the same: the ratios between classifier performances remain constant but performance is severely damaged as we increase the proportion of unknown examples to be tested (see Figure 6). Estimating the prior probability of finding an unknown example is a problem that has important implications for performance estimation in real scenarios, but this is not the only catch here. There is an artifact due to the performance criterion we use. F1 shows a tradeoff between recall and precision. Adding more negatives to the test set will keep recall constant while it can only damage precision. This effect must be taken into account when using F1 or related metrics to measure the performance in a context where we only vary the number of negatives.

There is also an obvious interaction between changing the number of negatives in the training sets while keeping constant the number of positive examples. For many supervised models, the recall is a monotonically non-increasing function of the number of negatives used to train them. The specificity is in turn a non-decreasing function of the number of training negatives and so this usually cancels out in measures like F1. The performance of the classifiers can appear similar but in fact the operating points may be

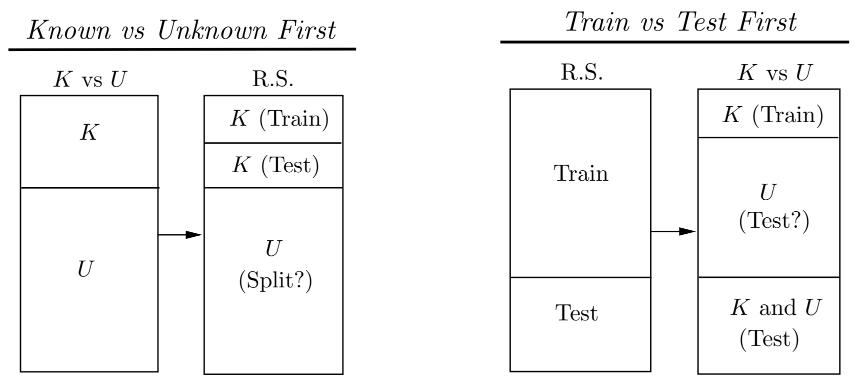


Fig. 5. Possible dataset splitting methodologies for research under the assumption of the existence of unknown labels. On the left, the original set is first split into known and unknown labels and then the examples from the known labels are sampled randomly (class stratification is difficult in multilabel datasets) to create training and testing sets. On the right, train and test sets are created first and then the training set is split into examples from the known and examples from the unknown labels; these approaches become roughly equivalent if the unknown-labels examples that do not participate in training are all added to the test set, with the exception that, due to the lack of stratification, the second approach does not guarantee the size of the actual training set.

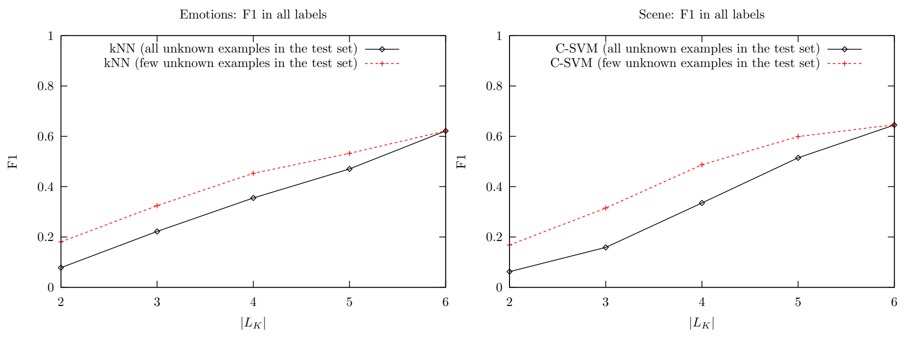


Fig. 6. Effect of increasing the number of the unknown examples in the test set. The F1 measure can only get worse, as recall remains constant while precision can only decrease. F1 measurements should be combined or replaced with other performance measures robust to this problem, such as the Balanced Accuracy Rate (average between recall and the true negative rate).

completely different, hiding the performance on the samples of interest, for example for the unknown examples. Again other measures must be used and tracking the evolution of both sides of the error becomes desirable.

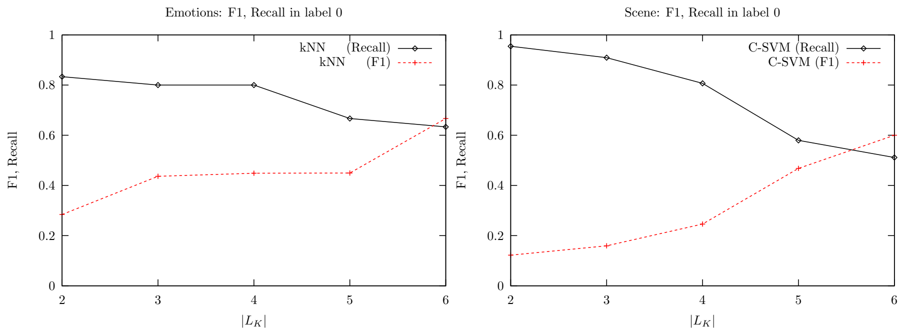


Fig. 7. Effect of increasing the number of negatives used for training an individual classifier. As we increase the number of negatives used for training the recall can only fall while, empirically, F1 remains constant or increases, due to both the classifiers rejecting more examples and getting less negatives in the testing sets. In an ideal scenario we would increase the proportion of positives accordingly, but our sample is finite. This is just another face of the problem with class skewness that must be taken into account in research with dynamic class sets.

5 Conclusions

In many ways research on multilabel classification is concerned with seeking solutions to the specific problems that arise on predicting individual labels. When reducing to binary classification this requires us to address problems like class skewness and high complexity of the concept space. In this respect, one-class classification is appealing, but its successful application would require a great deal of skill from the data miner. Choosing the right model, performing parameter selection and using heterogeneous combinations by choosing those models that are better adapted to the characteristic of each class are crucial tasks.The results in this paper are preliminary and should not be taken as of discouraging further research in this direction. Applying hybrid methods for introducing discriminative biases into one-class classifiers promises improvements. Apart from focusing on the individual classifiers, other previously proposed approaches, like splitting the label set to simplify the classification problems, appear to us as the most sensible lines of research in this area.

Research on scenarios where not all the classes are known at training time is not an active topic in multilabel classification at the moment. If it becomes relevant, a consensus on the proper evaluation methodology should be achieved, as different experimental setups can convey numerous artifacts. Here we conducted an initial attempt and stated some of the pitfalls we are aware of, but more reflection and debate are needed.

References

1. Tsoumakas, G., Katakis, I., Vlahavas, I.: Mining multi-label data. In Maimon, O., Rokach, L., eds.: Data Mining and Knowledge Discovery Handbook, 2nd edition. Springer (2009) in press
2. Beygelzimer, A., Langford, J., Zadrozny, B.: Machine learning techniques - reductions between prediction quality metrics. In: Performance Modeling and Engineering. Springer (2008) 3–28
3. Japkowicz, N.: The class imbalance problem: Significance and strategies. In: ICAI: International Conference on Artificial Intelligence. (2000)
4. Godbole, S., Sarawagi, S.: Discriminative methods for multi-labeled classification. In: PAKDD: Pacific-Asia Conference on Knowledge Discovery and Data Mining. (2004)
5. Spyromitros, E., Tsoumakas, G., Vlahavas, I.: An empirical study of lazy multilabel classification algorithms. In: Hellenic conference on Artificial Intelligence: Theories, Models and Applications. (2008)
6. Tax, D.M.J.: One-class classification. Concept learning in the absence of counterexamples. PhD thesis, Delft University of Technology (2001)
7. Raskutti, B., Kowalczyk, A.: Extreme re-balancing for SVMs: a case study. SIGKDD Explorations Newsletter **6**(1) (2004) 60–69
8. Tax, D.M.J., Duin, R.P.W.: Growing a multi-class classifier with a reject option. Pattern Recognition Letters **29**(10) (July 2008) 1565–1570
9. Lee, K., Kim, D.W., Na, D., Lee, K.H., Lee, D.: PLPD: reliable protein localization prediction from imbalanced and overlapped datasets. Nucleic acids research **34**(17) (2006) 4655–4666
10. Laskov, P., Düssel, P., Schäfer, C., Rieck, K.: Learning intrusion detection: Supervised or unsupervised? In: ICIAP: International Conference on Image Analysis and Processing. (2005)
11. Trohidis, K., Tsoumakas, G., Kalliris, G., Vlahavas, I.: Multilabel classification of music into emotions. In: ISMIR: International Conference on Music Information Retrieval. (2008)
12. Boutell, M.R., Luo, J., Shen, X., Brown, C.M.: Learning multi-label scene classification. Pattern Recognition **37**(9) (2004) 1757–1771
13. Elisseeff, A., Weston, J.: A kernel method for multi-labelled classification. In: NIPS: Advances in Neural Information Processing Systems. (2001)
14. Katakis, I., Tsoumakas, G., Vlahavas, I.: Multilabel text classification for automated tag suggestion. In: Proceedings of the ECML/PKDD 2008 Discovery Challenge. (2008)
15. 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: MULTIMEDIA: ACM International Conference on Multimedia. (2006)
16. Harmeling, S., Dornhege, G., Tax, D.M.J., Meinecke, F., Muller, K.R.: From outliers to prototypes: Ordering data. Neurocomputing **69**(13-15) (August 2006) 1608–1618
17. Schölkopf, B., Platt, J.C., Shawe-Taylor, J.C., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Computation **13**(7) (2001) 1443–1471
18. Platt, J.C.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In: Advances in Large Margin Classifiers. (1999)