Technische Universität Darmstadt
Knowledge Engineering Group
Hochschulstrasse 10, D-64289 Darmstadt, Germany

http://www.ke.informatik.tu-darmstadt.de

Technical Report TUD–KE–2008–04

Sang-Hyeun Park, Johannes Fürnkranz

Multi-Label Classification with Label Constraints

Multi-Label Classification with Label Constraints

Sang-Hyeun Park PARK@KE.INFORMATIK.TU-DARMSTADT.DE
Johannes Fürnkranz JUFFI@KE.INFORMATIK.TU-DARMSTADT.DE
Knowledge Engineering Group
Department of Computer Science
TU Darmstadt, Germany

Abstract

We extend the multi-label classification setting with constraints on labels. This leads to two new machine learning tasks: First, the label constraints must be properly integrated into the classification process to improve its performance and second, we can try to automatically derive useful constraints from data. In this paper, we experiment with two constraint-based correction approaches as post-processing step within the *ranking by pairwise comparison* (RPC)-framework. In addition, *association rule learning* is considered for the task of label constraints learning. We report on the current status of our work, together with evaluations on synthetic datasets and two real-world datasets.

1. Introduction

Multi-label classification is the problem of assigning a subset of all possible labels to each instance. Based on a number of training instances for which this association is known, the learning task is to infer a function that predicts a suitable subset of relevant labels for a new instance. This tasks occurs frequently in applications areas like text categorization (Lewis, 1997; Lewis et al., 2004; Menćia and Fürnkranz, 2008a) multimedia classification (Boutell et al., 2004; Snoek et al., 2006), or bioinformatics (Elisseeff and Weston, 2001; Diplaris et al., 2005). An excellent survey of the field can be found in (Tsoumakas and Katakis, 2007), a variety of resources, including benchmark datasets, at `http://mlkd.csd.auth.gr/multilabel.html`.

A straight-forward approach for addressing multi-label classification is to model each class independently. In the *binary relevance* approach, one binary classifier is trained for each possible label, in which all training examples for which the label is relevant are used as positives examples and all other examples as negative examples. However, in most real-world applications the predicted labels are not independent, so that the presence of one label may be indicative for other labels.

For this reason, several authors have extended the binary relevance approach to allow for incorporating dependencies between labels. For example, Crammer and Singer (2002) have proposed a training scheme for a binary relevance classifier that does not optimize the 0/1-loss of each individual label, but instead optimizes a given ranking loss function over the entire one-against-all ensemble. Menćia and Fürnkranz (2008b) have shown that this approach is outperformed by training a one-against-one ensemble, i.e., by having one classifier for each pair of labels.

In many applications, there are explicit constraints that must hold between the labels. For example, in the context of hierarchical classification, the presence of one label in the hierarchy often also implies the presence of all its ancestors. This situation can be modeled by a *subset constraint*, which specifies that whenever label λ_i is predicted as relevant, we must also predict λ_j. Similarly, one can imagine *exclusion constraints* specifying that two labels λ_i and λ_j cannot be relevant at the same time. A typical example of this case would be if the possible set of labels contains labels of several orthogonal dimensions, each having a set of mutually exclusive labels.

In this paper, we make two contributions: first, we will formally define the problem of multi-label learning with constraints and demonstrate the potential of this scenario on a simulated application with known constraints (Section 3). Second, we will evaluate an automated approach for discovering possible constraints on several well-known multi-label datasets (Section 4). Interestingly, we will see that in the automated approach, our results are mostly negative and cannot live up to the demonstrated potential on the artificial datasets.

2. Pairwise Multi-Label Classification

We will start with a brief recapitulation of multi-label classification, of pairwise classification techniques for tackling multi-label ranking and multi-label classification problems, and of multi-label and ranking loss functions.

2.1 Multi-Label Classification

Let $L = \{\lambda_i \,|\, i = 1 \ldots m\}$ be a set of labels and $X = \{x_k \,|\, k = 1 \ldots n\}$ be a set of instances, which are typically represented as feature vectors. Each instance x_k is associated with a set of relevant labels $L^+_{x_k} \subseteq L$. The set of irrelevant labels is denoted as $L^-_{x_k} = L \backslash L^+_{x_k}$. If it is clear from the context, that the same instance x_k is addressed, we will omit the index for convenience.

The task of multi-label classification is to find a function $f : X \to 2^L$, which takes as input an instance x and returns a set of labels $\tilde{L}^+_x \subseteq L$ as output. This function f should minimize the empirical risk for some loss function $l : 2^L \times 2^L \to \mathbb{R}^+_0$.

$$\sum_{x \in X} l(f(x),\, L^+_x) = \sum_{x \in X} l(\tilde{L}^+_x,\, L^+_x)$$

2.2 Multi-Label Ranking by Pairwise Comparison

In this work, we tackle the multi-label classification task within the *ranking by pairwise comparison* (RPC) framework (Hüllermeier et al., 2008), which addresses problems with structured output spaces by decomposing the problem into pairwise binary classifiers, one for each pair of classes, and combining their predictions, typically by a voting-based aggregation scheme. We assume the simplest scenario, in which each of the pairwise subproblems $\{p_{i,j} \,|1 \leq i < j \leq m\}$ returns a simple binary prediction $p_{i,j} : X \to \{0, 1\}$, which is interpreted as an unweighted vote for one of its two classes. These votes can be interpreted as *pairwise preferences*, i.e., the prediction $p_{i,j} = 1$ may be interpreted as a preference statement $\lambda_i \succ \lambda_j$, which means that label λ_i is preferred over λ_j. Conversely, the prediction $p_{i,j} = 0$ is interpreted as $\lambda_j \succ \lambda_i$.

At classification time, we will combine the predictions of the base classifiers $p_{i,j}$ with simple unweighted voting. Let $t = \frac{m(m-1)}{2}$ be the number of pairwise preferences respectively base classifiers for m labels, and let $\mathcal{S}_m$ be the space of all permutations of L.[1] Function $a : \{0, 1\}^t \to \mathcal{S}_m$ is the so called aggregation function, which combines all results of the decomposed subproblems to one result. Given $p_{i,j}$ and a, function f has the following form:

$$f := g(a(\{p_{i,j}\})) \tag{1}$$

where $g(.)$ is a threshold function that separates the label ranking into relevant and irrelevant labels.

Let $v : L \to \mathbb{N}^+_0$ be the function which counts the *votes* of one label λ_i.

$$v(\lambda_i) := \sum_{i<j\leq m} p_{i,j} + \sum_{1\leq j<i} (1 - p_{j,i}) \tag{2}$$

then voting aggregation is defined by:

$$a(\{p_{i,j}\}) = \tau = (\lambda^*_1, \lambda^*_2, \ldots, \lambda^*_m) \text{ with } v(\lambda^*_i) \geq v(\lambda^*_{i+1}),\ \forall i = 1 \ldots m-1 \tag{3}$$

1. Note that we will synonymously use the terms permutation and ranking, so $\mathcal{S}_m$ is also the space of all rankings consisting of L.

2.3 Calibrated Label Ranking

To convert the resulting ranking of labels into a multi-label prediction, we use the *calibrated label ranking* approach (Brinker et al., 2006, 2008). This technique avoids the need for learning a threshold function $g : \mathcal{S}_m \rightarrow L^+$ for separating relevant from irrelevant labels, which is often performed as a post-processing phase after computing a ranking of all possible classes. The key idea is to introduce an artificial *calibration label* λ_0, which represents the split-point between relevant and irrelevant labels. Thus, it is assumed to be preferred over all irrelevant labels, but all relevant labels are preferred over λ_0. As it turns out, the resulting m additional binary classifiers $\{\, p_{i,0} \,|\, i = 1 \dots m\}$ are identical to the classifiers that are trained by the binary relevance approach. Thus, each classifier $p_{i,0}$ is trained in a one-against-all fashion by using the whole dataset with $\{\, x \,|\, i \in L_x^+ \} \subseteq X$ as positive examples and $\{\, x \,|\, i \in L_x^- \} \subseteq X$ as negative examples. At prediction time, we will thus get a ranking over $m+1$ labels (the m original labels plus the calibration label). Then, the projection of voting aggregation of pairwise preferences with a calibrated label to a multi-label output is quite straight-forward:

$$\tilde{L}^+ = \{\lambda \,|\, v(\lambda) > v(\lambda_0)\} \subseteq L_c$$

or

$$g((\lambda_1^*, \lambda_2^*, \dots, \lambda_m^*)) := \{\lambda_i^* \,|\, i < \tau(\lambda_0)\}$$

where $\tau : L \rightarrow \mathbb{N}_0^+$ is a function which returns for a given label λ its position within the ranking τ.

2.4 Ranking and Multi-Label Loss Functions

In order to evaluate the predicted calibrated label-ranking we use different ranking and multi-label losses. The losses are computed comparing the ranking with the true set of relevant labels, each of them focusing on different aspects. For a given instance x, a relevant label set L^+, a negative label set $L^- = L \backslash L^+$ and a given predicted ranking τ the different loss functions are computed as follows:

RankingErr The normalized ranking error loss returns the normalized sum of squared position differences for each label in the predicted and true ranking. It is 0 for a ranking which is identical to the true ranking and 1 for a complete reversed ranking. Let τ^* be the true ranking.

$$l_{\text{RankingErr}} = \sum_{\lambda \in L} |\tau^*(\lambda) - \tau(\lambda)|^2$$

This loss corresponds to the Spearman rank correlation between two rankings.

ErrSetSize The error set size loss returns the number of pairs of labels which are not correctly ordered.

$$E = \{(\lambda, \lambda') \,|\, \tau(\lambda) > \tau(\lambda')\} \subseteq L^+ \times L^-$$
$$l_{\text{ErrSetSize}} = |E|$$

It corresponds to Kendall's τ, which measures the correlation between two rankings.

Margin The margin loss returns the number of positions between the worst ranked relative and the best ranked irrelevant label. This is directly related to the number of wrongly ranked labels, i.e. the relevant labels that are ordered below a irrelevant label, or vice versa. We denote this set by F.

$$F = \{(\lambda \in L^+ \,|\, \tau(\lambda) > \tau(\lambda'), \lambda' \in L^-\} \cup \{(\lambda' \in L^- \,|\, \tau(\lambda) > \tau(\lambda'), \lambda' \in L^+\}$$
$$l_{\text{Margin}} = \max(0, \max\{\tau(\lambda) \,|\, \lambda \in L^+\} - \min\{\tau(\lambda') \,|\, \lambda' \notin L^+\})$$

AvgP Average precision is commonly used in *information retrieval* and computes for each relevant label the percentage of relevant labels among all labels that are ranked before it, and averages these percentages over all relevant labels. In order to bring this loss in line with the others so that an optimal ranking is 0, we revert the measure.

$$l_{\text{AvgP}} = 1 - \frac{1}{|L^+|} \sum_{\lambda \in L^+} \frac{|\{\lambda^* \in L^+ \mid \tau(\lambda^*) \leq \tau(\lambda)\}|}{\tau(\lambda)}$$

3. Multi-Label Classification with Label Constraints

In this section, we describe the definition of constraints, and define straight-forward algorithms for correcting predictions that violate these constraints.

3.1 Definition of Label Constraints

In addition to the ordinary multi-label classification setting, we assume that we are given a set of *constraints* $C = \{c_i \mid i = 1 \ldots p\}$ on the labels L. In this paper, we consider two types of constraints: *subset* and *exclusion* constraints.

Subset constraints $\lambda_i \rhd \lambda_j$ denote that if label λ_i is relevant for a given instance x than λ_j has to be also relevant. Formally,

$$\lambda_i \rhd \lambda_j := \lambda_i \in L^+ \rightarrow \lambda_j \in L^+ \tag{4}$$

Exclusion constraints $\lambda_i \parallel \lambda_j$ denote that for all instances, labels λ_i and λ_j exclude each other, i.e., the two labels cannot be relevant or irrelevant at the same time. Formally,

$$\lambda_i \parallel \lambda_j := (\lambda_i \in L^+ \leftrightarrow \lambda_j \in L^-) \vee (\lambda_i \in L^- \leftrightarrow \lambda_j \in L^+) \tag{5}$$

We call subset or exclusion constraints *pairwise* if they have only one label on each side of their rule, and denote the space of all pairwise constraints for a given set of labels L as $\mathbb{C}_2(L)$.

There are several other ways to define constraint types on labels for the multi-label setting. For example, one could also consider the following four types of constraints:

$$\begin{aligned}
\lambda_i \rhd \lambda_j &:= \lambda_i \in L^+ \rightarrow \lambda_j \in L^+ \\
\lambda_i \blacktriangleright \lambda_j &:= \lambda_i \in L^- \rightarrow \lambda_j \in L^- \\
\lambda_i \overline{\rhd} \lambda_j &:= \lambda_i \in L^+ \rightarrow \lambda_j \in L^- \\
\lambda_i \overline{\blacktriangleright} \lambda_j &:= \lambda_i \in L^- \rightarrow \lambda_j \in L^+
\end{aligned}$$

Combined with logical connectors, these four basic constraints can represent a wide variety of constraints. For example, an exclusion constraint $\lambda_i \parallel \lambda_j$ may be viewed as a disjunction of the four constraints $(\lambda_i \overline{\rhd} \lambda_j) \vee (\lambda_i \overline{\blacktriangleright} \lambda_j) \vee (\lambda_j \overline{\rhd} \lambda_i) \vee (\lambda_j \overline{\blacktriangleright} \lambda_i)$

Such constraints are quite similar to instance-level constraints that have been explored in semi-supervised or constraint-based clustering (Wagstaff and Cardie, 2000; Wagstaff et al., 2001; Bilenko et al., 2004), only that we define constraints between different labels (known groups of instances), whereas the constraints for semi-supervised clustering are defined between instances (e.g., this pair of instances must (not) belong to the same cluster).

3.2 Constraint-Based Correction of Predictions

Basically, label constraints can be integrated into the learning phase or testing phase of multi-label classification. Within the RPC framework, pairwise subset constraints like $\lambda_i \rhd \lambda_j$ could be easily modelled in the learning phase, i.e. by substituting the pairwise preference $p_{1,2}$ with a constant function, which returns always 1. Therefore, this specific preference woulds always prefer λ_j and would not have to be learned anymore. However, this approach does not guarantee that the constraint is respected in the final prediction, because individual preferences may be over-ridden in the aggregation phase. Thus, we focus on integrating label constraints into the aggregation phase, where the predictions of the individual classifiers are combined.

Hence, we interpret the given constraints as immutable *hard* constraints, which must be respected by the final multi-label prediction. In addition, the predicted pairwise preferences are interpreted as *soft* constraints, which should be respected as well, but may be violated if necessary. These altering should be minimal for some distance measure. We consider two possible measures. First, the number of preference swappings that are needed to make the predicted preferences conform to the final prediction, and second, the number of neighboring label swappings in the predicted ranking. Our algorithm starts with an invalid predicted ranking and searches for a valid ranking which can be constructed by a minimal amount of preference or neighbor label swappings.

Algorithm 1: PrefSwap

```
Input: Constraints $C$, pairwise preferences $P$, invalid ranking $\tau_0$
Output: a set of valid rankings best with minimum number of preference swappings

best = {} ;
candidates = {$\tau_0$};
evaluated = {};
repeat
    newCandidates = {};
    foreach $c \in$ candidates do                          /* expand new rankings */
        foreach $p \in P$ do                           /* by iterating preferences */
            $\tau_{\text{new}} = c$.swapPreference($p$);
            if $\tau_{new} \notin$ evaluated then            /* enqueue only new rankings */
                newCandidates = newCandidates $\cup$ {$\tau_{\text{new}}$};
    foreach $\tau \in$ newCandidates do                        /* check constraints */
        if $\tau$ is valid then
            best = best $\cup$ {$\tau$};
    evaluated = evaluated $\cup$ candidates $\cup$ newCandidates;
    candidates = newCandidates;
until best $\neq$ {} or newCandidates = {} ;
```

3.2.1 Minimizing Preference Swappings (PS)

The preference swappings measure is motivated by the assumption that an invalid ranking is caused by a few incorrectly predicted pairwise preferences. Errors among the pairwise base classifiers are assumed to be independent. If we denote $\mathcal{S}_m(C)$ as all permutations respectively rankings with $m = |L|$ which satisfy C, and denote with P the set of pairwise preferences, then our distance measure can be formulated as:

$$d_{\text{PS}} = |P| - \max_{P_1 \in \mathcal{S}_m} |P_1 \cap P| \quad \text{with } a(P_1) \in \mathcal{S}_m(C)$$

Our implementation of finding a PS-minimal ranking is based on *breadth-first search*, and is presented as pseudocode in algorithm 1. We start with an invalid predicted ranking τ_0. For every possible pairwise preference $p \in \{p_{i,j} \mid 1 \leq i < j \leq m\}$, the ranking τ_{new} is generated, which yields by swapping followed by voting-aggregation. Then, to avoid multiple checks of the same ranking, only new rankings are appended to the newCandidates queue. After this expanding step, the candidate

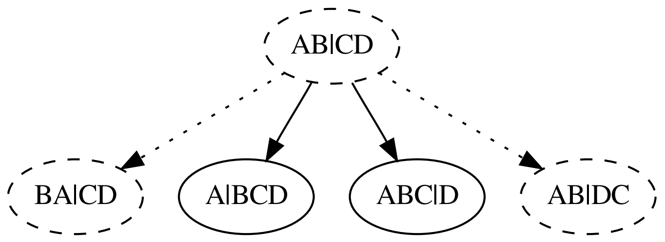

Figure 1: A simple example of constraint correction by neighbor label swapping. The predicted invalid ranking is $(AB|CD)$ and the constraint set consists of only one element : $B \rhd C$. NLS expands the initial ranking and returns two valid rankings. Dashed nodes represent invalid and solid nodes valid rankings.

rankings $\tau_n \in \mathsf{newCandidates}$ are checked if any satisfy the given constraints. If one ranking is determined as valid, the search process does not stop but all remaining rankings in the queue will still be processed. We refer to this scheme in the further text as PS. Note that the elements in newCandidates represent rankings of the same level, the actual highest depth. So, all rankings $\tau \in \mathsf{newCandidates}$, which satisfy the constraints, are equal in terms of swapped preferences. If the PS-minimal ranking is not unique further selection criterions are evaluated, which are described later.

3.2.2 Minimizing Neighbor-Label Swappings (NLS)

Neighbor label swapping is motivated by the fact that swapping one preference yields at most to a swapping of two labels in the ranking, whose position difference is 1. We refer these label pairs as neighboring or adjacent. However, many swappings of individual preferences will not yield a change in the predicted ranking. So as an approximation, one can use the needed swappings of neighbor or adjacent labels as a minimizing criteria. In another view, minimizing NLS directly relates to one valid ranking with minimal RankingErr to the predicted ranking. If we denote τ_0 as the predicted ranking and $k : \mathcal{S}_m \times m-1 \rightarrow \mathcal{S}_m$ as the swapping function $k((\lambda_1, \dots, \lambda_m), n) = (\lambda_1, \dots, \lambda_{n-1}, \lambda_{n+1}, \lambda_n, \dots, \lambda_m)$ then the measure can be formulated as:

$$d_{\mathrm{NLS}} = \arg\min_{n \in \mathbb{N}_0^+} < y_n > \quad \text{with } k(\dots k(k(\tau_0, y_1), y_2) \dots, y_n) \in \mathcal{S}_m(C)$$

The algorithm to minimize Neighbor-Label Swappings (NLS) is a straight-forward adaption of PrefSwap, which iterates through all neighboring labels rather than through all pairwise preferences. Note that this scheme has a linear (in the number of labels) branching factor and PrefSwap a quadratic one.

3.2.3 Comparing and Tie-Breaking

Given $d_{\min} = \min_{\tau_i \in \mathcal{S}_m} d_{PS}(\tau_0, \tau_i)$ and several valid rankings τ_j with $d_{PS}(\tau_0, \tau_j) = d_{\min}$, we choose RankingErr as first criterion to further distinguish among them. This is consistent with the objective to minimize the changes of the initial invalid predicted ranking to satisfy some given constraints. NLS-minimized rankings omit this step, since they are by construction equal with respect to RankingErr.

Table 1: Experiments on synthetic data generated with constraints C_1.

error	RankingErr			ErrSetSize			AvgP			# viol.
	VA	PS	NLS	VA	PS	NLS	VA	PS	NLS	
0.05	0.025	**0.024**	0.026	0.055	**0.048**	0.053	0.008	**0.007**	0.007	0.10
0.10	0.053	**0.052**	0.054	0.171	**0.165**	0.166	0.025	0.025	**0.024**	0.17
0.15	**0.085**	0.086	0.088	0.329	**0.318**	0.321	0.043	0.041	**0.041**	0.23
0.20	**0.125**	0.126	0.128	0.556	**0.545**	0.548	0.072	**0.070**	0.070	0.27
0.25	**0.168**	0.169	0.171	0.813	0.800	**0.799**	**0.080**	0.097	0.096	0.31
0.30	**0.227**	0.227	0.228	1.251	1.235	**1.235**	0.144	0.142	**0.140**	0.34

Table 2: Experiments on synthetic data generated with constraints C_2.

error	RankingErr			ErrSetSize			AvgP			# viol.
	VA	PS	NLS	VA	PS	NLS	VA	PS	NLS	
0.05	0.023	**0.022**	0.022	0.045	0.043	**0.041**	0.008	0.007	**0.007**	0.06
0.10	0.052	0.050	**0.049**	0.170	0.160	**0.153**	0.028	0.027	**0.026**	0.13
0.15	0.085	0.082	**0.082**	0.368	0.342	**0.340**	0.055	0.053	**0.052**	0.19
0.20	0.123	0.119	**0.118**	0.609	0.584	**0.576**	0.083	0.081	**0.080**	0.24
0.25	0.169	0.163	**0.163**	0.936	0.894	**0.891**	0.118	0.114	**0.114**	0.29
0.30	0.223	0.217	**0.215**	1.312	1.279	**1.265**	0.159	0.157	**0.155**	0.34

There are cases in which this criterion is still equal for some rankings, see for example figure 1. Suppose the predicted ranking[2] is $\tau_0 = (AB|CD)$ and a domain expert has specified the constraint $c_1 = B \rhd C$. The ranking τ_0 does not satisfy c_1. It can be trivially repaired by swapping the position of the calibration label | with one of its neighbors B or C, yielding the $\tau_1 = (A|BCD)$ and $\tau_2 = (ABC|D)$. Both are equal with respect to the NLS distance. Two other rankings, $(BA|CD)$ and $(AB|DC)$, can also be found at the same search depth, but these are invalid.

In order to be able to decide for one of the two valid rankings, we first compute the RankingErr with respect to the originally predicted ranking τ_0. If this also equal between the candidates (as in our example), we check if the direct neighbors of the predicting split point violate the initial pairwise preferences. Let $s_p = \tau_i(\lambda_0)$ be the position of the splitpoint within a ranking τ_i, then we compute:

$$|\mathcal{P} \cap \{\lambda_0 \succ \tau_i^{-1}(s_p - 1),\ \lambda_0 \prec \tau_i^{-1}(s_p + 1)\}|$$

So in other words, we count the number of wrongly ordered neighbor label pairs which are direct above or below the splitpoint. We select the one with the lowest number. Then if there are still ambiguous rankings, we select the one which minimizes the disordered number for all $m-1$ neighbor label pairs, not only the direct neighbors of the splitpoint. As a last separation step, a random selection is applied.

3.3 Experimental Evaluation

In the following, we show the results of the PS and NLS algorithms on artificial data.

2. We will use the notation $\tau_0 = (AB|CD)$, where the labels are ordered according to decreasing level of relevance (A is most relevant, D is least relevant), and the splitpoint between relevant and irrelevant labels is indicated with a "|".

Table 3: Experiments on 100 random synthetic datasets. For each loss function the left values are generated by ordinary voting-aggregation. The right values show constraint-correction values based on neighbor-label swapping.

ERROR	RankingErr		ErrSetSize		Margin		AvgP		# VIOL.
	VA	NLS	VA	NLS	VA	NLS	VA	NLS	
0.05	0.0062	**0.0059**	0.0602	**0.0488**	0.0601	**0.0488**	0.0074	**0.0065**	0.11
0.10	0.0134	**0.0126**	0.1868	**0.1562**	0.1862	**0.1554**	0.0223	**0.0196**	0.20
0.15	0.0218	**0.0207**	0.3910	**0.3393**	0.3868	**0.3350**	0.0449	**0.0402**	0.27
0.20	0.0322	**0.0307**	0.6743	**0.6053**	0.6602	**0.5903**	0.0750	**0.0686**	0.33
0.25	0.0441	**0.0423**	1.0256	**0.9462**	0.9894	**0.9080**	0.1098	**0.1024**	0.37
0.30	0.0579	**0.0561**	1.4603	**1.3765**	1.3813	**1.2930**	0.1519	**0.1439**	0.41

3.3.1 Data Generation

Given a set of labels $L = \{1 \dots m\}$ and a set of pairwise label constraints $C \subseteq \mathbb{C}(L)_2$, n random permutations $\tau_1, \dots, \tau_n \in \mathcal{S}_m$ are generated, which satisfy C. Each of the permutation τ is decomposed into the unique set $P = \{p_{i,j} \,|\, 1 \leq i < j \leq m\}$ of binary pairwise preferences. For example, if $\tau = (\lambda_1, \lambda_3, \lambda_2)$ then $P = \{\lambda_1 \succ \lambda_2, \lambda_1 \succ \lambda_3, \lambda_2 \prec \lambda_3\}$ is the associated set of binary pairwise preferences. The classification error of the binary pairwise classifiers is modeled by swapping a ratio (ERROR $= 0.05, 0.1, 0.15, 0.2, 0.25, 0.3$) of the pairwise preferences.

3.3.2 Evaluation

In a first experiment, we used two arbitrarily chosen constraint sets on 6 labels ($C_1 = \{1 \rhd 2, 3 \rhd 4, 4 \rhd 5, 6 \rhd 5\}$ and $C_2 = \{1 \parallel 2, 2 \parallel 5, 3 \rhd 6\}$), and generated $n = 5000$ training instances for each.

Table 1 and 2 show the results of the comparison between regular voting aggregation (VA), and the constraint-based corrections PS and NLS. For each loss function the values in the leftmost column are generated by voting-aggregation without any constraint-based post-correction. The second and third column show constraint-correction values based on preference swapping and neighbor label swapping. The bold numbers describe the best values for a particular loss and ERROR combination. Margin error values are omitted for lack of space. Their relations among the different aggregations schemes are anyway mostly identical to the ErrSetSize values, more precisely, the aggregation scheme with the best Margin value for a particular ERROR is identical to the best one for ErrSetSize. For both set of constraints C_1 and C_2, PS or NLS tend to outperform VA, but the results are not entirely conclusive.

To obtain a more thorough evaluation, we used 100 datasets with random rankings for 6 labels, each with 5000 instances. The number of constraints was also randomly selected from 2 to 5. These constraints were first checked for consistency and finally evaluated for six ERROR values ($\epsilon = 0.05, 0.1, 0.15, 0.2, 0.25, 0.3$). The average losses and ratios of violated instances are shown in table 3. In this evaluation, only NLS was used as correction scheme, since its evaluation takes significantly less time and its performance seem to be competitive to PS. The values clearly show the superior performance of NLS-minimizing constraint correction compared to simple voting-aggregation. For each ERROR - loss function combination NLS outperforms the baseline.

4. Discovering Label Constraints from Data

In many domains, sensible label constraints may be available from background knowledge about the target domain. However, even in domains in which such knowledge is not readily available, one may try to automatically discover the knowledge from data. In this section, we evaluate the use of association rule learning algorithms for this purpose.

4.1 Association Rules as Constraints

We define the problem of discovering label constraints in the data as an *association rule learning* problem. We construct one item set for each training example x_i, which consists of the set of relevant labels L_i^+. We then use a association rule learner to discover rules of the form

$$\lambda_{i_1} \ldots \lambda_{i_b} \rightarrow \lambda_j$$

with b labels in the antecedent and one label in the consequent. Negation can be handled by including negative labels of the form $-\lambda_k$ with the semantic $\lambda_k \in L^-$ into the itemsets. Thus, each example is associated with an itemset of length m, one item for each label denoted either as λ_k or $-\lambda_k$.

Typical association rule learning algorithms tend to generate redundant rules. These are justified in their original main application areas, i.e. market basket analysis, since their main goal is to find (all) interesting rules or relations between items rather than a compact set of rules. However, for our purpose, to use association rules as constraints, these redundant rules lead to unnecessary runtime growth. In this work we understand redundancy in the sense of inductive rule learning. We are thus interested in generating rules with minimal antecedent, as opposed to, e.g., closed itemset mining which considers rules with maximal antecedent (Goethals, 2005).

A rule $A \rightarrow B$ consisting of body (antecedent) A and head (consequent) B is *redundant* with respect to rule $C \rightarrow B$ if C is a subset of A. If a rule is more *specific* than another, it is unnecessary to check, because the more general rule will be checked in any case. So in our evaluations we speed up the constraint correction process, by post-processing generated association rules with a *minimizing* step, which removes all rules except the most general ones. In the above example, if $C \subsetneq A$, then the rule $A \rightarrow B$ will be removed.

4.2 Experiments on Real-World Data

We compare simple voting aggregation and the constraint correction algorithm on two real-world multi-label datasets, namely *yeast* and *siam*.[3] The dataset *yeast* consists of 14 labels, 1500 training and 917 testing instances. It concerns the functional multi-label classification of yeast genes. Dataset *siam* is a text-categorization problem, where multiple labels are associated to one document. It consists of 22 labels, 21519 training and 7077 testing data. We used the given training / test splits for evaluation.

The association rules were generated by the APriori algorithm (Agrawal et al., 1996) in its implementation by Borgelt (2003). As a base learner, we used the support-vector machine implementation in LIBSVM (Chang and Lin, 2001) with a linear kernel in its default settings. The algorithms were compared according to the same metrics as above, except that we cannot give RankingErr values, since we did not have correct rankings of the datasets to compute this loss function.

Table 4 shows the result of the evaluation on the *yeast* dataset. The values in the first line represent performance values for aggregation of pairwise preferences by voting, which is used as our baseline. The next lines, beginning with various minimum confidence and support values, describe the result of NLS constraint correction with different sets of constraints, which are generated by association rule learning using stated parameters on the training data. The last column describes the amount of violated instances, and therefore the number of instances to which constraint correction was applied. In all other cases, the predicted ranking was not changed. APriori with parameters $c = 90$ and $s = 20$ generated inconsistent rules, so no corresponding values are shown.

3. The datasets are available online at http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel.html.

Table 4: Experiments on Real-World Data : *yeast*. The right-most column shows the amount and the ratio of predicted rankings which the violated given constraint set.

Conf	Supp	ErrSetSize	Margin	AvgP	# violated
VA		6.4602	4.3533	0.2426	
100	60	6.4602	4.3533	0.2426	28 (0.03)
	40	6.4558	4.3511	0.2425	102 (0.11)
	20	6.4667	4.3544	0.2430	303 (0.33)
95	60	6.4591	4.3533	0.2426	39 (0.04)
	40	6.4569	4.3490	0.2425	111 (0.12)
	20	6.4667	4.3479	0.2429	341 (0.37)
90	60	6.4591	4.3533	0.2426	40 (0.04)
	40	6.4547	4.3479	0.2426	174 (0.19)
	20	-	-	-	-

Table 5: Experiments on real-world data : *siam*. The right-most column shows the amount and the ratio of predicted rankings which violated the given constraint set.

Conf	Supp	ErrSetSize	Margin	AvgP	# violated
VA		1.7254	1.5947	0.1920	
100	60-20	1.7254	1.5947	0.1920	2 (0.00)
95	90-70	1.7375	1.6062	0.1967	1157 (0.16)
90	95	1.7349	1.6044	0.1958	768 (0.11)
	90	1.7409	1.6094	0.1978	1926 (0.27)
	85	1.7396	1.6079	0.1969	2205 (0.31)
	80-70	1.7455	1.6141	0.1985	2609 (0.37)

As one can see, constraint correction with association rules as constraints does not cause significant changes in the performance of multi-label classification. Even in cases where a considerable amount of instances had to be post-processed, for example $c = 95$ and $s = 20$, where 37% of the predicted rankings violated some of the learned constraints, no real difference to the baseline can be observed. The results for *siam* (Table 5) even show a consistent deterioration in prediction performance, i.e., for all applications of constraint correction the evaluated losses are worse or equal than the baseline.

Some performance values for *siam* in Table 5 are identical for different support values with the same confidence, i.e. $c = 100$, $s = 50$ and $c = 100$, $s = 30$. This is caused by the fact, that identical association rules were generated for these parameters. More information regarding the used association rules as constraints can be seen in Tables 6 and 7, which show the number of generated constraints for the varying confidence and support values. In addition, the rightmost column shows the number of rules, which survived our crude redundancy filter, and were (as previously described) actually used in the constraint testing process.

Table 6: Constraint Generation: *yeast*

Conf	Supp	# rules	# min
100	60	65	8
	40	735	11
	20	11321	46
95	60	245	21
	40	2067	33
	20	27042	99
90	60	305	31
	40	2398	44
	20	31708	127

Table 7: Constraint Generation: *siam*

Conf	Supp	# rules	# min
100	60	8	1
	50	2957	3
	40	35041	3
	30	168882	3
	20	466284	6
95	90	2296	143
	80	52204	191
	70	324442	198
90	95	109	70
	90	2416	182
	85	15905	239
	80	61652	273
	75	178920	281
	70	415861	288

5. Discussion

We introduced constraints into the multi-label classification setting, and studied two machine learning tasks in this context:

1. Integration of additional knowledge in form of label constraints into the multi-label classification setting
2. Automatically learning of label constraints

Regarding the first point, we experimented with two approaches which tackle the constraint integration problem by transforming it into a search problem - searching for a valid ranking with minimal distance from the ordinary predictions. The number of preference swappings (PS) and the number of neighbor-label swappings (NLS) seem to be intuitive and reasonable choices as distance functions within the RPC framework. Although empirical evaluations of PS and NLS on artificial datasets showed a improvement for multi-label classification, it failed for two commonly used real-world datasets, where we used automatically discovered constraints.

In our view, several points could be the reason for the negative results. At first, one could criticize that we had given the correct constraints for the artificial datasets, which was not the case for the real-world datasets. One is that the introduced setting with given true constraints may be too idealistic. Indeed, for our two evaluated real-world datasets, we have no evidence, even for rules with $c = 100$ that these rules hold for all instances from the true distribution, since the rules were generated on training data, which might differ from the true distribution. Small tests with association rules generated with parameters $c = 100$, $s = 1$ on training and testdata of *yeast* showed also no improvement.

Another point is that the artificial data was explicitly modelled by voting *deaggregation* - given transitive (binary) pairwise preferences, the correct calibrated label-ranking is uniquely defined and vice-versa (if we exclude ties). Pairwise preferences in general do not have to be transitive.

Besides the failure on real-world data, we are aware that the shown algorithms are currently not applicable to practical problems. We perform an essentially exhaustive breadth-first search through all possible rankings, and also use a rather expensive pruning step for the association rule discovery. Without strong assumptions, i.e. that a valid ranking is relatively fast reachable by PS or NLS for an invalid ranking, the search process takes too long, since the number of possible candidates grows exponentially for each iteration of the search algorithm. More efficient algorithms are currently under investigation.

However, our main goal was to investigate whether this approach can, in principle, yield improved results. Despite the negative results with automatically discovered constraints, we nevertheless interpret our results as informative, and plan a deeper investigation of this learning scenario.

Acknowledgments

We would like to thank Tom Mitchell for suggesting the problem of multi-label classification with constraints. This research was supported by the *German Science Foundation (DFG)*.

References

Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. *Advances in knowledge discovery and data mining*, pages 307–328, 1996.

Mikhail Bilenko, Sugato Basu, and Raymond J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the 21st International Conference on Machine Learning (ICML-04)*, page 11, New York, NY, USA, 2004. ACM.

Cristian Borgelt. Efficient implementations of Apriori and Eclat. In *Proceedings of the 1st Workshop of Frequent Item Set Mining Implementations (FIMI-03)*, Melbourne, FL, USA, 2003.

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

Klaus Brinker, Johannes Fürnkranz, and Eyke Hüllermeier. A unified model for multilabel classification and ranking. In Gerhard Brewka, Silvia Coradeschi, Anna Perini, and Paolo Traverso, editors, *ECAI*, pages 489–493. IOS Press, 2006.

Klaus Brinker, Johannes Fürnkranz, Eyke Hüllermeier, and Eneldo Loza Menćia. Multilabel classification via calibrated label ranking. *Machine Learning*, 2008. In Press.

Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Koby Crammer and Yoram Singer. A new family of online algorithms for category ranking. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 151–158, New York, NY, USA, 2002. ACM.

Sotiris Diplaris, Grigorios Tsoumakas, Pericles A. Mitkas, and Ioannis P. Vlahavas. Protein classification with multiple algorithms. In Panayiotis Bozanis and Elias N. Houstis, editors, *Panhellenic Conference on Informatics*, volume 3746 of *Lecture Notes in Computer Science*, pages 448–456. Springer, 2005.

André Elisseeff and Jason Weston. A kernel method for multi-labelled classification. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *NIPS*, pages 681–687. MIT Press, 2001.

Bart Goethals. Frequent set mining. In *In The Data Mining and Knowledge Discovery Handbook*, chapter 17, pages 377–397. Springer, 2005.

Eyke Hüllermeier, Johannes Fürnkranz, Weiwei Cheng, and Klaus Brinker. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 2008. To Appear.

David D. Lewis. *Reuters-21578 text categorization test collection*, 1997. README file (V 1.2), available from http://www.research.att.com/~lewis/reuters21578/README.txt.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

Eneldo Loza Menćia and Johannes Fürnkranz. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. Springer-Verlag, 2008a.

Eneldo Loza Menćia and Johannes Fürnkranz. Pairwise learning of multilabel classifications with perceptrons. In *Proceedings of the IEEE World Congress on Computational Intelligence (WCCI-08)*. IEEE, 2008b.

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.

Grigorios Tsoumakas and Ioannis Katakis. Multi label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13, 2007.

Kiri Wagstaff and Claire Cardie. Clustering with instance-level constraints. In P. Langley, editor, *Proceedings of the 17th International Conference on Machine Learning (ICML-2000)*, pages 1103–1110, Stanford, CA, 2000. Morgan Kaufmann.

Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings of the 18th International Conference on Machine Learning (ICML-01)*, pages 577–584, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.