Efficient Voting Prediction for Pairwise Multilabel Classification

Eneldo Loza Mencía, Sang-Hyeun Park, Johannes Fürnkranz

TU Darmstadt - Knowledge Engineering Group
Hochschulstr. 10 - Darmstadt - Germany

Abstract

The pairwise approach to multilabel classification reduces the problem to learning and aggregating preference predictions among the possible labels. A key problem is the need to query a quadratic number of preferences for making a prediction. To solve this problem, we extend the recently proposed *QWeighted* algorithm for efficient pairwise multiclass voting to the multilabel setting, and evaluate the adapted algorithm on several real-world datasets. We achieve an average-case reduction of classifier evaluations from n^2 to $n + dn \log n$, where n is the total number of possible labels and d is the average number of labels per instance, which is typically quite small in real-world datasets.

Key words: multilabel classification, voting aggregation, learning by pairwise comparison, efficient classification

1. Introduction

Multilabel classification refers to the task of learning a function that maps instances $\bar{\mathrm{x}} \in \mathcal{X}$ to label subsets $P_{\bar{\mathrm{x}}} \subset \mathcal{L}$, where $\mathcal{L} = \{\lambda_1, \ldots, \lambda_n\}$ is a finite set of predefined labels, typically with a small to moderate number of alternatives. Thus, in contrast to multiclass learning, alternatives are not assumed to be mutually exclusive, such that multiple labels may be associated with a single instance.

A prototypical application scenario for multilabel classification is the assignment of a set of keywords to a document, a frequently encountered problem in

Email addresses: eneldo@ke.tu-darmstadt.de (Eneldo Loza Mencía), park@ke.tu-darmstadt.de (Sang-Hyeun Park), juffi@ke.tu-darmstadt.de (Johannes Fürnkranz)

This is the authors' version of the work retrieved from www.ke.tu-darmstadt.de. The original publication appeared in *Neurocomputing*, 73:7-9, pp. 1164 – 1176, 2010, doi:10.1016/j.neucom.2009.11.024, and is available at www.sciencedirect.com/science/article/B6V10-4Y646JK-3/2/6c043e816e88387deed6ff0301a9666c

the text classification domain. With upcoming Web 2.0 technologies this domain is extended by a wide range of tag suggestion tasks (e.g. (Tsoumakas, Katakis, and Vlahavas, 2008; Katakis, Tsoumakas, and Vlahavas, 2008)). This kind of problems are often associated with a large number of instances or classes which demand for an efficient processing. The Reuters-2000 dataset for instance is composed of over 800,000 documents and 103 classes (cf. Section 5.2), a benchmark extracted from the *del.icio.us* platform contains almost 1000 classes (Tsoumakas et al., 2008) and the EUR-Lex database consists of almost 4000 classes (cf. Section 5.2). Other tasks include protein classification and semantic multimedia annotation.

The predominant approach to multilabel classification is *binary relevance learning* (BR). In BR the problem is decomposed into several binary problems in the following way: for each class a binary classifier is trained to discriminate between examples of the class and the examples of the remaining classes. A different approach is to have a classifier for each possible pair of classes that is trained to distinguish only between these two classes. This approach is usually denominated *one-vs-one*, *round robin* or *pairwise classification* and has shown to achieve a higher predictive quality in the multiclass (Fürnkranz, 2002; Hsu and Lin, 2002) as well as in the multilabel case (Loza Mencía and Fürnkranz, 2008b; Fürnkranz, Hüllermeier, Loza Mencía, and Brinker, 2008).

It has been shown that the complexity for training an ensemble of pairwise classifiers is comparable to the complexity of training a BR ensemble (Fürnkranz, 2002; Loza Mencía and Fürnkranz, 2008b). The reason is that even though we have a quadratic number of classifiers in a pairwise ensemble as opposed to a linear number in the BR ensemble, each of the pairwise classifiers contains fewer examples. More precisely, each original training example occurs in all of the n BR classifiers, whereas it only occurs in $n-1$ pairwise classifiers. Thus, the total number of training examples in the union of all training sets is smaller in the pairwise case. The fact that these examples are distributed over a larger number of different classifiers makes the pairwise approach particularly attractive for expensive classifiers like SVMs.

However, the problem remains that a quadratic number of classifiers has to be evaluated to produce a prediction. Our first attempts in efficient multilabel pairwise classification lead to the algorithm MLPP, which uses the fast perceptron algorithm as base classifier. With this algorithm, we successfully tackled the large Reuters-RCV1 text classification benchmark, despite the quadratic number of base classifiers (Loza Mencía and Fürnkranz, 2008b). Although we were able to beat the competing fast MMP algorithm (Crammer and Singer, 2003) in terms of ranking performance and were competitive in training time, the costs for testing were not satisfactory.

Park and Fürnkranz (2007) recently introduced a method named *QWeighted* for multiclass problems that intelligently selects only the base classifiers that are actually necessary to predict the top class. This reduced the evaluations needed from $n(n-1)/2$ to only $n \log n$ in practice, which is near the n evaluations processed by BR.

In this paper we introduce a novel algorithm which adapts the *QWeighted*

method to the MLPP algorithm. In a nutshell, the adaption works as follows: instead of stopping when the top class is determined, we repeatedly apply *QWeighted* to the remaining classes until the final label set is predicted. In order to determine at which position to stop, we use the calibrated label ranking technique (Fürnkranz et al., 2008), which introduces an artificial label for indicating the boundary between relevant and irrelevant classes. We evaluated this technique on a selection of multilabel datasets that vary in terms of problem domain, number of classes and label density. The results demonstrate that our modification allows the pairwise technique to process such data in comparable time to the one-per-class approaches while producing more accurate predictions.

Nevertheless, this novel algorithm still uses a quadratic number of base classifiers, i.e. the memory requirements grow quadratically to the number of classes. This problem cannot be solved by the *QWeighted* approach, however, we have recently introduced a variant of MLPP that represents the perceptrons as a linear combination of training examples in order to enable the algorithm to handle a large amount of classes (Loza Mencía, Park, and Fürnkranz, 2008; Loza Mencía and Fürnkranz, 2008b).

A related work on the issue of efficient multilabel classification is the *HOMER* algorithm by Tsoumakas et al. (2008). The label set is organized through clustering into a hierarchy of labels. A multilabel classifier is then trained at each inner node. This reformulating leads to less complex problems at each inner node and hence allows to train the classifier ensemble more efficiently in terms of computations and memory. We are currently cooperating with the authors on combining *HOMER* and our approach in order to reduce the memory requirement and obtained first encouraging results (Tsoumakas, Katakis, Loza Mencía, Park, and Fürnkranz, 2009).

This work is organized as follows: Section 2 defines the problem and describes the basic algorithms such as perceptrons, binary relevance, MLPP and CMLPP. Section 3 introduces *QWeighted* and the adaptation to multilabel classification. In Section 4 we compare the time and space complexity of the different algorithms. Section 5 is dedicated to the experimental setup along with the used datasets and evaluation measures, the results are presented in Section 6. Section 7 provides a final discussion and concludes this paper.

2. Multilabel Classification

We represent an instance or object as a vector $\bar{\mathrm{x}} = (x_1, \ldots, x_a)$ in a feature space $\mathcal{X} \subseteq \mathbb{R}^a$. Each instance $\bar{\mathrm{x}}_i$ is assigned to a set of relevant labels P_i, a subset of the n possible classes $\mathcal{L} = \{\lambda_1, \ldots, \lambda_n\}$. For multilabel problems, the cardinality $|P_i|$ of the label sets is not restricted, whereas for binary problems $|P_i| = 1$. For the sake of simplicity we use the following notation for the binary case: we define $\mathcal{L} = \{1, -1\}$ as the set of classes so that each object $\bar{\mathrm{x}}_i$ is assigned to a class $\lambda_i \in \{1, -1\}$, $P_i = \{\lambda_i\}$.

2.1. Perceptrons

We use the simple but fast perceptrons as base classifiers (Rosenblatt, 1958). As support vector machines (SVM), their decision function describes a hyperplane that divides the a-dimensional space into two halves corresponding to positive and negative examples. We use a version that works without learning rate and threshold:

$$o(\bar{\mathrm{x}}) = sgn(\bar{\mathrm{x}} \cdot \bar{\mathrm{w}}) \tag{1}$$

with the internal weight vector $\bar{\mathrm{w}}$ and $sgn(t) = 1$ for $t \geq 0$ and -1 otherwise. If there exists a *separating hyperplane* between the two sets of points, i.e. they are linearly separable, it is proved that the following update rule finds it (cf., e.g., (Bishop, 1995)).

$$\alpha_i = (\lambda_i - o(\bar{\mathrm{x}}_i)) \qquad\qquad \bar{\mathrm{w}}_{i+1} = \bar{\mathrm{w}}_i + \alpha_i \bar{\mathrm{x}}_i \tag{2}$$

The main reason for choosing the perceptrons as our base classifier is because, contrary to SVMs, they can be trained efficiently in an incremental setting, which makes them particularly well-suited for large-scale classification problems such as the Reuters-RCV1 benchmark (Lewis, Yang, Rose, and Li, 2004), without forfeiting too much accuracy though SVMs find the *maximum-margin hyperplane* (Freund and Schapire, 1999; Crammer and Singer, 2003; Shalev-Shwartz and Singer, 2005).

In addition, important advancements were achieved in recent times trying to adapt the perceptron algorithm in order to maximize the margin of the separating hyperplane, without losing the advantages of simplicity and efficiency that characterize the perceptron algorithm (Li, Zaragoza, Herbrich, Shawe-Taylor, and Kandola, 2002; Crammer, Dekel, Keshet, Shalev-Shwartz, and Singer, 2006; Khardon and Wachman, 2007; Tsampouka and Shawe-Taylor, 2007). The presented algorithms can easily be adapted in order to use these variants if desired.

Nevertheless, we also experimented with SVMs as base classifier. Although we were able to increase prediction accuracy in some cases, many datasets could not be processed despite using the efficient LibLinear library (Fan, Chang, Hsieh, Wang, and Lin, 2008).

2.2. Binary Relevance Ranking

In order to provide a baseline and to show the efficiency of the pairwise approach, we compare our algorithms to the binary relevance (BR) or one-against-all (OAA) variant with perceptrons as base classifier.

In the binary relevance method, a multilabel training set with n possible classes is decomposed into n binary training sets that are then used to train n binary classifiers. So for each pair $(\bar{\mathrm{x}}_i, P_i)$ in the original training set n different pairs $(\bar{\mathrm{x}}_i, \lambda_{i_j})$ with $j = 1 \ldots n$ are generated as follows:

$$\lambda_{i_j} = \begin{cases} 1 & \lambda_j \in P_i \\ -1 & otherwise \end{cases} \tag{3}$$

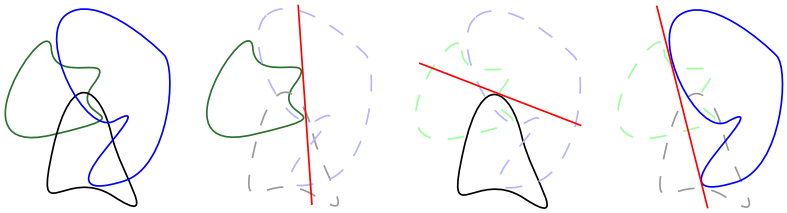


Figure 1: Subproblems in *binary relevance* for multilabel classification: original three-class problem (green, blue and black classes, shown as overlapping clouds in left picture) is divided into green vs. rest (second picture), black vs. rest (third) and blue vs. rest two-class subproblems. Separating hyperplanes, denoted by red lines, have to respect all examples (inside the clouds). Clouds of negative examples have dotted lines.

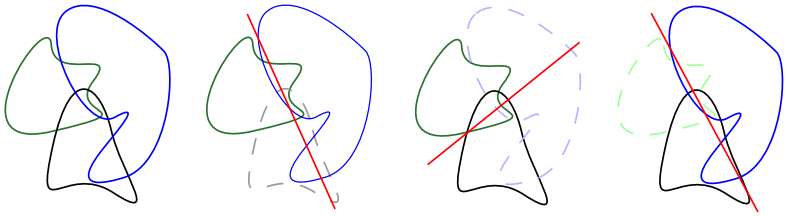


Figure 2: Subproblems in *pairwise multilabel classification*: original three-class problem is divided into green vs. blue (second picture, black examples are ignored), green vs. black (blue is ignored) and blue vs. black two-class subproblems. Separating hyperplanes have to respect only examples from two classes in contrast to BR in Figure 1. Dotted lines denote the ignored class.

Note, that all of these n decomposed training sets are of the same size as the original training set. A brief visual description of this technique is available in Figure 1.

Supposing we use perceptrons as base learners, n different o_j classifiers are trained in order to determine the relevance of λ_j. In consequence, the combined prediction of the binary relevance classifier for an instance $\bar{\mathrm{x}}$ would be the set $\{\lambda_j \mid o_j(\bar{\mathrm{x}}) = 1\}$. If, in contrast, we want to obtain a ranking of classes according to their relevance, we can simply use the result of the internal computation of the perceptrons as a measure of relevance. According to (1) the desired linear combination is the inner product $o'_j(\bar{\mathrm{x}}) = \bar{\mathrm{x}} \cdot \bar{\mathrm{w}}_j$. So the result of the prediction is a vector $\bar{\mathrm{o}}'(\bar{\mathrm{x}}) = (\bar{\mathrm{x}}\bar{\mathrm{w}}_1, \ldots, \bar{\mathrm{x}}\bar{\mathrm{w}}_n)$ where component j corresponds to the relevance of class λ_j. Ties are broken randomly to not favor any particular class.

2.3. Multilabel Pairwise Perceptrons (MLPP)

In the pairwise binarization method for multiclass classification, one classifier is trained for each pair of classes, i.e., a problem with n different classes is decomposed into $\frac{n(n-1)}{2}$ smaller subproblems. For each pair of classes (λ_u, λ_v), only examples belonging to either λ_u or λ_v are used to train the corresponding

Require: Training example pair $(\bar{\mathrm{x}}_i, P_i)$, perceptrons $\{\bar{\mathrm{w}}_{u,v} \mid u < v, \lambda_u, \lambda_v \in \mathcal{L}\}$
1: $N_i \leftarrow \mathcal{L} \setminus P_i$
2: **for each** $(\lambda_u, \lambda_v) \in P_i \times N_i$ **do**
3: **if** $u < v$ **then**
4: $\bar{\mathrm{w}}_{u,v} \leftarrow \textsc{TrainPerceptron}(\bar{\mathrm{w}}_{u,v}, (\bar{\mathrm{x}}_i, 1))$ ▷ train as positive example
5: **else**
6: $\bar{\mathrm{w}}_{v,u} \leftarrow \textsc{TrainPerceptron}(\bar{\mathrm{w}}_{v,u}, (\bar{\mathrm{x}}_i, -1))$ ▷ train as negative example
7: **return** $\{\bar{\mathrm{w}}_{u,v} \mid u < v, \lambda_u, \lambda_v \in \mathcal{L}\}$ ▷ updated perceptrons

Figure 3: Pseudocode of the (incremental) training method of the MLPP algorithm.

$o_{1,2} = 1$	$o_{2,1} = -1$	$o_{3,1} = -1$	$o_{4,1} = -1$	$o_{5,1} = -1$
$o_{1,3} = 1$	$o_{2,3} = 1$	$o_{3,2} = -1$	$o_{4,2} = -1$	$o_{5,2} = -1$
$o_{1,4} = 1$	$o_{2,4} = 1$	$o_{3,4} = 1$	$o_{4,3} = -1$	$o_{5,3} = -1$
$o_{1,5} = 1$	$o_{2,5} = 1$	$o_{3,5} = 1$	$o_{4,5} = 1$	$o_{5,4} = -1$
$v_1 = 4$	$v_2 = 3$	$v_3 = 2$	$v_4 = 1$	$v_5 = 0$

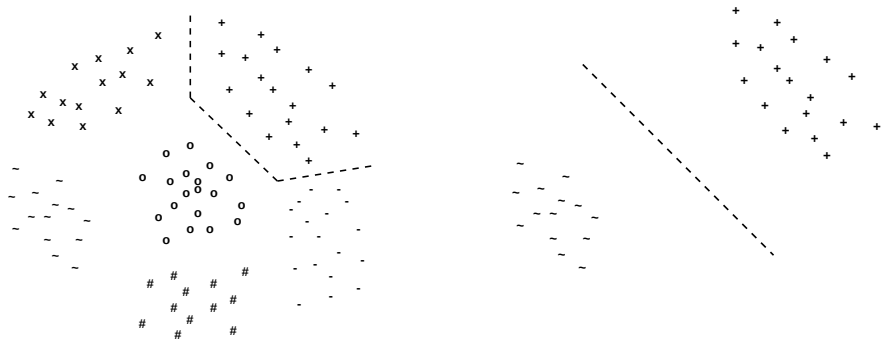
Figure 4: MLPP voting: an example $\bar{\mathrm{x}}$ is classified by all 10 base perceptrons $o_{i,j}, i \neq j,\ \lambda_i, \lambda_j \in \mathcal{L}$. Note the redundancy given by $o_{i,j} = -o'_{j,i}$. The last line counts the positive outcomes for each class.

classifier $o_{u,v}$. All other examples are ignored. In the multilabel case, and assuming $u < v$, an example is added to the training set for classifier $o_{u,v}$ if λ_u is a relevant class and λ_v is an irrelevant class or vice versa, i.e., if $(\lambda_u, \lambda_v) \in P_i \times N_i$ or vice versa $(\lambda_u, \lambda_v) \in N_i \times P_i$ with $N_i = \mathcal{L} \setminus P_i$ as negative labelset (cf. Figure 2). Thus training examples of class λ_u will receive a training signal of $+1$, whereas training examples of class λ_v will be classified with -1. Figure 3 shows the training algorithm in pseudocode. Of course, MLPPs being perceptrons, they can also be trained incrementally.

During classification, the predictions of the base classifiers $o_{u,v}$ are interpreted as *preference statements* that predict for a given example which of the two labels λ_u or λ_v is preferred. In order to convert these binary preferences into a class ranking, we use a simple voting strategy known as *max-wins*, which interprets each binary preference as a vote for the preferred class. Classes are then ranked according to the number of received votes after the evaluation of all $\frac{n(n-1)}{2}$ perceptrons. Ties are broken randomly in our case.

Figure 4 shows a possible result of classifying the sample instance of Figure 6. Perceptron $o_{1,5}$ predicts (correctly) the first class, consequently λ_1 receives one vote and class λ_5 zero (denoted by $o_{1,5} = 1$ in the first and $o_{5,1} = -1$ in

Several extensions of the pairwise approach, such as Pairwise Correcting Classifiers (Moreira and Mayoraz, 1998) and Tri-Class SVMs (Angulo, Ruiz, González, and Ortega, 2006), also integrate the remaining examples into the training process. In several experiments, this has lead to an improved performance, which has to be paid with a considerable increase in training time, and more complex decision boundaries for the involved classifiers.



(a) *binary-relevance classification* n classifiers, each separates one class from all other classes. Here: + against all other classes.

(b) *pairwise classification* $\frac{n(n-1)}{2}$ classifiers, one for each pair of classes. Here: + against $\sim$.

Figure 5: One-against-all and pairwise binarization.

the last row). All 10 perceptrons (the values in the upper right corner can be deduced due to the symmetry property of the perceptrons) are evaluated though only six are 'competent' since only they were trained with the original example.

This may be disturbing at first sight since many non-competent perceptrons, i.e. perceptrons discriminating λ_u and λ_v for a given instance $\bar{\mathrm{x}}_i$ with $\lambda_u, \lambda_v \in P_i$ or $\lambda_u, \lambda_v \in N_i$, are involved in the voting process: $o_{1,2}$ is asked though it cannot know anything relevant in order to determine if $\bar{\mathrm{x}}$ belongs to λ_1 or λ_2 since it was neither trained on this example nor on other examples belonging simultaneously to both classes λ_1 and λ_2 (or to none of both). In the worst case the resulting noisy votes (votes from non-competent perceptrons) concentrate on a single negative class, which would lead to misclassifications. But note that any class can at most receive $n-1$ votes, so that in the extreme case when the competent perceptrons all classify correctly and the non-competent ones concentrate on a single class, a positive class would still receive at least $n-|P|$ and a negative at most $n-|P|-1$ votes. Class λ_3 in Figure 4 is an example for this: It receives all possible noisy votes but still loses against the positive classes λ_1 and λ_2.

The pairwise binarization method is often regarded as superior to binary relevance because it profits from simpler decision boundaries in the subproblems (Fürnkranz, 2002; Hsu and Lin, 2002). In the case of an equal class distribution, the subproblems have $\frac{2}{n}$ times the original size whereas binary relevance maintains the size. Typically, this goes hand in hand with an increase of the space where a separating hyperplane can be found. An intuitive visualization of this aspect can be found in Figure 5 for the multiclass case and in Figure 2 for the multilabel case, in contrast to the BR binarization depicted in Figure 1. A simple example also illustrates this: imagine you repeatedly insert points around two points on a line. The distance between the two sets will inevitably

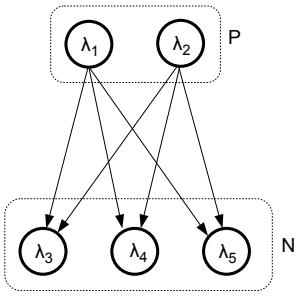


Figure 6: MLPP training: training example $\bar{\mathbf{x}}$ belongs to $P_{\bar{\mathbf{x}}} = \{\lambda_1, \lambda_2\}$, $N_{\bar{\mathbf{x}}} = \{\lambda_3, \lambda_4, \lambda_5\}$ are the irrelevant classes, the arrows represent the trained perceptrons $\bar{\mathbf{w}}_{1,3}, \bar{\mathbf{w}}_{1,4}, \bar{\mathbf{w}}_{1,5}, \bar{\mathbf{w}}_{2,3}, \bar{\mathbf{w}}_{2,4}, \bar{\mathbf{w}}_{2,5}$.

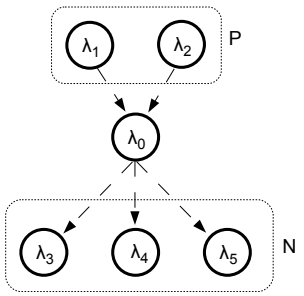


Figure 7: calibration: introducing virtual label λ_0 that separates P and N. Perceptrons $\bar{\mathbf{w}}_{1,0}, \bar{\mathbf{w}}_{2,0}, \bar{\mathbf{w}}_{0,3}, \bar{\mathbf{w}}_{0,4}, \bar{\mathbf{w}}_{0,5}$ are additionally trained.

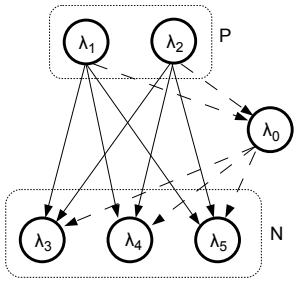


Figure 8: CMLPP training: the complete set of trained perceptrons.

monotonically decrease with increasing number of points. Thus it is very likely for a subproblem to have a larger margin than the full problem.

Particularly in the case of text classification the obtained benefit clearly exists. An evaluation of the pairwise approach on the Reuters-RCV1 corpus (cf. Section 5.2), which contains over 100 classes and 800,000 documents, showed a significant and substantial improvement over the MMP method (Loza Mencía and Fürnkranz, 2008b).

2.4. Calibrated Label Ranking

To convert the resulting ranking of labels into a multilabel prediction, we use the *calibrated label ranking* approach (Brinker, Fürnkranz, and Hüllermeier, 2006; Fürnkranz et al., 2008). This technique avoids the need for learning a threshold function 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

$o_{0,1} = -1$	$o_{1,0} = 1$	$o_{2,0} = 1$	$o_{3,0} = -1$	$o_{4,0} = -1$	$o_{5,0} = -1$
$o_{0,2} = -1$	$o_{1,2} = 1$	$o_{2,1} = -1$	$o_{3,1} = -1$	$o_{4,1} = -1$	$o_{5,1} = -1$
$o_{0,3} = 1$	$o_{1,3} = 1$	$o_{2,3} = 1$	$o_{3,2} = -1$	$o_{4,2} = -1$	$o_{5,2} = -1$
$o_{0,4} = 1$	$o_{1,4} = 1$	$o_{2,4} = 1$	$o_{3,4} = 1$	$o_{4,3} = -1$	$o_{5,3} = -1$
$o_{0,5} = 1$	$o_{1,5} = 1$	$o_{2,5} = 1$	$o_{3,5} = 1$	$o_{4,5} = 1$	$o_{5,4} = -1$
$v_0 = 3$	$v_1 = 5$	$v_2 = 4$	$v_3 = 2$	$v_4 = 1$	$v_5 = 0$

Figure 9: MLPP voting with calibrated label λ_0: an example $\bar{\mathrm{x}}$ is classified by all 15 base perceptrons. The last line counts the positive outcomes for each class.

assumed to be preferred over all irrelevant labels, but all relevant labels are preferred over λ_0. This introduction of an additional label during training is depicted in Figure 7, the combination with the normal pairwise base classifiers is shown in Figure 8.

As it turns out, the resulting n additional binary classifiers $\{\, o_{i,0} \mid i = 1 \ldots n\}$ are identical to the classifiers that are trained by the binary relevance approach. Thus, each classifier $o_{i,0}$ is trained in a one-against-all fashion by using the whole dataset with $\{\, \bar{\mathrm{x}} \mid \lambda_i \in P_{\bar{\mathrm{x}}}\} \subseteq \mathcal{X}$ as positive examples and $\{\, \bar{\mathrm{x}} \mid \lambda_i \in N_{\bar{\mathrm{x}}}\} \subseteq \mathcal{X}$ as negative examples. At prediction time, we will thus get a ranking over $n + 1$ labels (the n original labels plus the calibration label). Then, the projection of voting aggregation of pairwise perceptrons with a calibrated label to a multilabel output is quite straight-forward:

$$\hat{P} = \{\lambda \in \mathcal{L} \mid v(\lambda) > v(\lambda_0)\}$$

where $v(\lambda)$ is the amount of votes class λ has received.

Figure 9 extends the example from figure 4 and shows a possible result of classifying with the calibrated label λ_0. It shows the ideal case, where for instance, the relevant classes λ_1 and λ_2 receive a vote, respectively, in direct comparison with the calibrated label (perceptrons $o_{1,0}$ and $o_{2,0}$). After evaluating all perceptrons, the number of votes for the calibrated label $v(\lambda_0) = v_0$ is used as the split-point to discriminate relevant classes from irrelevant classes. In this example, λ_1 and λ_2 are returned as the set of relevant classes $\hat{P}$.

We denote the MLPP algorithm adapted in order to support the calibration technique as CMLPP. This algorithm was again applied to the large Reuters-RCV1 corpus and to other smaller datasets presented also in Section 5.2, outperforming the binary relevance and MMP approach. For further details please refer to (Fürnkranz et al., 2008).

3. Quick Weighted Voting

As already seen, the quadratic number of base classifier does not seem to be a serious drawback for training MLPP and also CMLPP. However, at prediction time it is still necessary to evaluate a quadratic number of base classifier. Two approaches to overcome this problem for multiclass and for multilabel task are presented in the following.

Require: example $\bar{\mathrm{x}}$; classifiers $\{o_{u,v} \mid u < v, \lambda_u, \lambda_v \in \mathcal{L}\}$; $l_1, \ldots, l_n = 0$

while λ_{top} not determined **do**
  $\lambda_a \leftarrow \mathrm{argmin}_{\lambda_i \in \mathcal{L}}\ l_i$ ▷ select top candidate class
  $\lambda_b \leftarrow \mathrm{argmin}_{\lambda_j \in \mathcal{L} \setminus \{\lambda_a\}}\ l_i$ **and** $o_{a,b}$ not yet evaluated ▷ select second
  if no λ_b exists **then**
    $\lambda_{top} \leftarrow \lambda_a$ ▷ top rank class determined
  else ▷ evaluate classifier
    $v_{ab} \leftarrow o_{a,b}(\bar{\mathrm{x}})$ ▷ one vote for λ_a ($v_{ab} = 1$) or λ_b ($v_{ab} = 0$)
    $l_a \leftarrow l_a + (1 - v_{ab})$ ▷ update voting loss for λ_a
    $l_b \leftarrow l_b + v_{ab}$ ▷ update voting loss for λ_b

Figure 10: Pseudocode of the *QWeighted* algorithm (multiclass classification).

3.1. QWeighted for Multiclass Classification

For the multiclass case, the simple but effective voting strategy, which is applied often to combine the predictions of pairwise classifiers to one multiclass classification result, can be computed efficiently with the Quick Weighted Voting algorithm (*QWeighted*) (Park and Fürnkranz, 2007), which is shown in Figure 10. Instead of the evaluation of the quadratic number of all pairwise perceptrons, it is possible to evaluate a smaller subset of it in order to compute the class with the highest accumulated voting mass.

During a voting procedure there exist many situations where particular classes can be excluded from the set of possible top rank classes, even if they reach the maximal voting mass in the remaining evaluations. Its main idea can be described in a simple example: Given n classes with $n > j$, if class λ_a has received more than $n - j$ votes and class λ_b lost j votings, it is impossible for λ_b to achieve a higher total voting mass than λ_a. Thus further evaluations with λ_b can be safely ignored for the comparison of these two classes.

Pairwise classifiers will be selected depending on a *loss* value, which is the amount of potential voting mass that a class has *not* received. More precisely, the loss l_i of a class λ_i is defined as $l_i := p_i - v_i$, where p_i is the number of evaluated incident classifiers of λ_i and v_i is the current vote amount of λ_i. Obviously, the loss will begin with a value of zero and is monotonically increasing. The class with the current minimal loss is one of the top candidates for the top rank class.

First the pairwise classifier $o_{a,b}$, in our case the perceptron $\bar{\mathrm{w}}_{a,b}$, will be selected for which the losses l_a and l_b of the relevant classes λ_a and λ_b are minimal, provided that the classifier $o_{a,b}$ has not yet been evaluated. In the case of multiple classes that have the same minimal loss, there exists no further distinction, and we select a class randomly from this set. Then, the losses l_a and l_b will be updated based on the evaluation returned by $o_{a,b}$ (recall that v_{ab} is interpreted as the amount of the voting mass of the classifier $o_{a,b}$ that goes to class λ_a and $1 - v_{a,b}$ is the amount that goes to class λ_b). These two steps will be repeated until all classifiers for the class λ_m with the minimal loss has been evaluated. Thus the current loss l_m is the correct loss for this class. As all

other classes already have a greater loss, λ_m is the correct *top rank* class.

Theoretically, a minimal number of comparisons of $n-1$ is possible (*best case*). The *worst case*, on the other hand, is still $n(n-1)/2$ comparisons, which can, e.g., occur if all pairwise classifiers classify randomly with a probability of 0.5. In practice, the number of comparisons will be somewhere between these two extremes, depending on the nature of the problem.

3.2. QWeighted for Multilabel Classification

A simple adaptation of *QWeighted* to multilabel classification is to repeat the process. We can compute the top class λ_{top} using *QWeighted* and remove this class from $\mathcal{L}$ and repeat this step, until the returned class is the artificial label λ_0, which means that all remaining classes will be considered to be irrelevant. This adaptation uses two simple extensions of the original algorithm. Firstly, the information about which pairwise perceptrons have been evaluated and their results are carried through the iterations so that no pairwise perceptron is evaluated more than once. And secondly, by using the calibrated label ranking approach we know beforehand that at some point the vote amount of the artificial label has to be computed. So, in hope for a *better* starting distribution of votes, all incident classifiers $o_{i,0}$ respectively $\bar{w}_{i,0}$ of the artificial label are evaluated explicitly before employing iterated *QWeighted*. We denote this method as QCMLPP1.

In addition to this straight-forward adaptation, we considered also a slightly improved variant (QCMLPP2). In retrospect, QCMLPP1 computes a partial ranking of classes down to the calibrated label. That means that for all relevant labels all their incident classifiers are evaluated. It neglects the fact that for multilabel classification the information that a particular class *is ranked above* the calibrated label is sufficient, rather than *to which amount*. QCMLPP2 works in the same way as QCMLPP1 except that it stops the evaluation of the current top rank λ_t if it already received a higher voting mass than the calibrated label. The class λ_t is not automatically removed from the set of labels as in QCMLPP1, since further evaluations for the computation of other classes can occur, but it can not be selected as a new top rank candidate. The pseudocode of QCMLPP2 is depicted in Fig. 11.

Note that the effectiveness of this testing procedure is highly influenced by the relation of average number of relevant labels to total number of labels. We can expect a high reduction of pairwise comparisons if the above relation is relatively small, which holds for the most real-world multilabel datasets.

We are currently investigating further variants for improving the performance. For example, different search heuristics based on other losses than the number of "lost games" are imaginable. Furthermore, the selection of the two next classes for evaluation can also be varied, i.e. by pairing the "best" and the "worst" class in the next iteration instead of the two currently best classes. In addition, we are working on the derivation of formal complexity bounds to strengthen the *QWeighted* approach.

Require: example $\bar{\mathrm{x}}$; classifiers $\{o_{u,v} \mid u < v, \lambda_u, \lambda_v \in \mathcal{L}\}$; $l_0, \ldots, l_n = 0$

$v_0 \leftarrow 0,\ \tilde{P} \leftarrow \emptyset$

for $i = 0$ to n **do** ▷ evaluate all classifiers of artificial label λ_0
  $l_i \leftarrow o_{0,i}(\bar{\mathrm{x}})$
  $v_0 \leftarrow v_0 + (1 - l_i)$ ▷ compute votes of calibrated label

repeat
  while λ_{top} not determined **do** ▷ apply adapted *QWeighted*
    $\lambda_a \leftarrow \mathrm{argmin}_{\lambda_i \in \mathcal{L}}\ l_i$
    $\lambda_b \leftarrow \mathrm{argmin}_{\lambda_j \in \mathcal{L} \setminus \{\lambda_a\}}\ l_i$ **and** $o_{a,b}$ not yet evaluated
    if $v_a \geq v_0$ **or** no λ_b exists **then** ▷ adapted stopping criterion
      $\lambda_{top} \leftarrow \lambda_a$
    else ▷ evaluate classifier
      $v_{ab} \leftarrow o_{a,b}(\bar{\mathrm{x}})$ ▷ update statistics
      $v_a \leftarrow v_a + v_{ab}$
      $v_b \leftarrow v_b + (1 - v_{ab})$
      $l_a \leftarrow l_a + (1 - v_{ab})$
      $l_b \leftarrow l_b + v_{ab}$

  if $v_{top} \geq v_0$ **then**
    $\tilde{P} \leftarrow \tilde{P} \cup \lambda_{top}$ ▷ relevant label found
    $l_{top} \leftarrow +\infty$ ▷ arrange λ_{top} at the end of possible opponents queue

until $v_{top} \geq v_0$ **and** $|\tilde{P}| < n$ ▷ check if all relevant labels found

return $\tilde{P}$ ▷ return relevant labels

Figure 11: Pseudocode of the QCMLPP2 algorithm.

4. Computational Complexity

The notation used in this section is the following: n denotes the number of possible classes, d the average number of relevant classes per instance in the training set, a the number of attributes and a' the average number of attributes not zero (size of the sparse representation of an instance), and m denotes the size of the training set. For each complexity we will give an upper bound O in Landau notation. We will indicate the runtime complexity in terms of real value additions and multiplications ignoring operations that have to be performed by all algorithms such as sorting or internal real value operations. Additionally, we will present the complexities per instance since all algorithms are incrementally trainable.

4.1. Memory Requirements

BR follows an one model per class approach, so it has to keep one perceptrons for each class in memory, leading to $O(n \cdot a)$ memory space. In contrast, the pairwise approaches require one perceptron for each of the $\frac{n(n-1)}{2}$ pairs of classes, hence we need $O(n^2 a)$ memory. In addition, the calibrated versions require an overhead of n perceptrons for the comparisons with the artificial label.

4.2. Training

For processing one training example, n dot products have to be computed by BR, plus at most the same amount if there was a prediction error. The MLPPs require $O(dn)$ dot products, one for each associated perceptron. Assuming that a dot product computation costs $O(a')$, we obtain a complexity of $O(dna')$ per training example. Thus, assuming similar loss rates, the pairwise training will be only on average d resp. $d + 1$ for the calibrated version slower than the BR algorithm despite training a quadratic number of base classifier.

4.3. Prediction

During prediction the one-per-class approach achieve $O(na')$ computations for one instance. For the pairwise approach without the usage of *QWeighted* all perceptrons have to be evaluated, leading to $O(n^2 a')$ computations. The same upper bound holds analytically for QCMLPP, but as previous experiments have shown for the multiclass case, *QWeighted* (QW) reduces the amount of required base classifier evaluations from $\frac{n(n-1)}{2}$ to $n \log(n)$ in practice (Park and Fürnkranz, 2007). Let C_{QW} be the runtime of one iteration of *QWeighted*. Then, it is easy to see that the number of base classifier evaluations for the multilabel adaptations of *QWeighted* is bounded from above by $n + d \cdot C_{\mathrm{QW}}$, since we always evaluate the n classifiers involving the calibrated class, and have to do one iteration of *QWeighted* for each of the (on average) d relevant labels. Assuming that *QWeighted* on average needs $C_{QW} = n \log(n)$ base classifier evaluations we can expect an average number of $n + dn \log n$ classifier evaluations for the QCMLPP variants, as compared to the $\approx n^2$ evaluations for

Table 1: Computational complexity given as upper bounds of number of addition and multiplication operations, for each instance. n: *#classes*, d: *avg. #labels per instance*, a: *#attributes*, a': *#attributes*$\neq 0$.

	training	prediction	memory
BR	$O(na')$	$O(na')$	$O(na)$
MLPP	$O(dna')$	$O(n^2a')$	$O(n^2a)$
QCMLPP	$O(dna')$	$\sim na' + dn\log(n)\ a'$	$O(n^2a)$

the regular CMLPP. Thus, the effectiveness of the adaption to the multilabel case crucially depends on the average number d of relevant labels. We can expect a high reduction of pairwise comparisons if d is small compared to n, which holds for most real-world multilabel datasets.

A compilation of the analysis can be found in Table 1, together with the complexities of BR. Note that the stated prediction time for QCMLPP in the table is not an analytical complexity bound like the others, it is an empirically estimated value.

At first view QCMLPP does not benefit analytically from the *QWeighted* voting, but there is empirical evidence for a clear improvement compared to the full voting. There is no disadvantage of using QCMLPP instead of CMLPP unless a more fine-grained distinction between classes than relevant-irrelevant is required.

Note that we have assumed a linear dependence on the number of training instances since we use the perceptron algorithm as our base classifier. For base classifiers with a super-linear relationship the ratio to BR in terms of training complexity may be further reduced due to the smaller subproblems (Fürnkranz, 2002). While a perceptron needs the same time for a problem of m examples than for n problems of $\frac{m}{n}$ examples, it does not hold the same for a learning algorithms like SVMs or C4.5 since $m^x > n(\frac{m}{n})^x = (\frac{m^x}{n^{x-1}})$ for $x > 1$.

5. Experimental Setup

5.1. Multilabel Evaluation Measures

There is no generally accepted procedure for evaluating multilabel classifications. Our approach is to consider a multilabel classification problem as a meta-classification problem where the task is to separate the set of possible labels into relevant labels and irrelevant labels. Let $\hat{P}_{\bar{\mathrm{x}}}$ denote the set of labels predicted by the multilabel classifier and $\hat{N}_{\bar{\mathrm{x}}} = \mathcal{L} \setminus \hat{P}_{\bar{\mathrm{x}}}$ the set of labels that are not predicted by the classifier. Thus, we can, for each individual instance $\bar{\mathrm{x}}$, compute a two-by-two confusion matrix $C_{\bar{\mathrm{x}}}$ of relevant/irrelevant vs. predicted/not predicted labels:

$C_{\bar{\mathbf{x}}}$	predicted	not predicted	
relevant	$\lvert P_{\bar{\mathbf{x}}} \cap \hat{P}_{\bar{\mathbf{x}}} \rvert$	$\lvert P_{\bar{\mathbf{x}}} \cap \hat{N}_{\bar{\mathbf{x}}} \rvert$	$\lvert P_{\bar{\mathbf{x}}} \rvert$
irrelevant	$\lvert N_{\bar{\mathbf{x}}} \cap \hat{P}_{\bar{\mathbf{x}}} \rvert$	$\lvert N_{\bar{\mathbf{x}}} \cap \hat{N}_{\bar{\mathbf{x}}} \rvert$	$\lvert N_{\bar{\mathbf{x}}} \rvert$
	$\lvert \hat{P}_{\bar{\mathbf{x}}} \rvert$	$\lvert \hat{N}_{\bar{\mathbf{x}}} \rvert$	$\lvert \mathcal{L} \rvert$

From such a confusion matrix $C_{\bar{\mathbf{x}}}$, we can compute several well-known measures:

- The *Hamming loss* (HamLoss) computes the percentage of labels that are misclassified, i.e., relevant labels that are not predicted or irrelevant labels that are predicted. This basically corresponds to the error in the confusion matrix.

$$\textsc{HamLoss}(C_{\bar{\mathbf{x}}}) \stackrel{\text{def}}{=} 1 - \frac{1}{|\mathcal{L}|}\left|\hat{P}_{\bar{\mathbf{x}}} \triangle P_{\bar{\mathbf{x}}}\right| \tag{4}$$

  The operator $\triangle$ denotes the symmetric difference between two sets and is defined as $A \triangle B \stackrel{\text{def}}{=} (A \setminus B) \cup (B \setminus A)$, i.e. $\hat{P}_{\bar{\mathbf{x}}} \triangle P_{\bar{\mathbf{x}}}$ has all labels that only appear in one of the two sets.

- *Precision* (Prec) computes the percentage of predicted labels that are relevant, *recall* (Rec) computes the percentage of relevant labels that are predicted, and the *F1*-measure is the harmonic mean between the two.

$$\textsc{Prec}(C_{\bar{\mathbf{x}}}) \stackrel{\text{def}}{=} \frac{|\hat{P}_{\bar{\mathbf{x}}} \cap P_{\bar{\mathbf{x}}}|}{|\hat{P}_{\bar{\mathbf{x}}}|} \qquad \textsc{Rec}(C_{\bar{\mathbf{x}}}) \stackrel{\text{def}}{=} \frac{|\hat{P}_{\bar{\mathbf{x}}} \cap P_{\bar{\mathbf{x}}}|}{|P_{\bar{\mathbf{x}}}|} \tag{5}$$

$$\text{F1}(C_{\bar{\mathbf{x}}}) \stackrel{\text{def}}{=} \frac{2}{\frac{1}{\textsc{Rec}(C_{\bar{\mathbf{x}}})} + \frac{1}{\textsc{Prec}(C_{\bar{\mathbf{x}}})}} = \frac{2\,\textsc{Rec}(C_{\bar{\mathbf{x}}})\,\textsc{Prec}(C_{\bar{\mathbf{x}}})}{\textsc{Rec}(C_{\bar{\mathbf{x}}}) + \textsc{Prec}(C_{\bar{\mathbf{x}}})} \tag{6}$$

To average these values, we compute a micro-average over all values in a test set, i.e., we add up the confusion matrices $C_{\bar{\mathbf{x}}}$ for examples in the test set and compute the measure from the resulting confusion matrix. Thus, for any given measure f and m test documents, the average is computed as:

$$f_{\text{avg}} = f(\sum_{i=1}^{m} C_i) \tag{7}$$

To combine the results of the individual folds of a cross-validation, we average the estimates f^j_{avg}, $j = 1 \ldots q$ over all q folds.

For the case of a zero denominator, a common convention is to define the result as zero.

Table 2: Statistics of datasets. The attribute number in parenthesis denotes the actual used number of features, i.e. for *scene* and *yeast* the number of features after adding the pairwise products and for the text collections the amount after feature selection. *Label-set size d* denotes the average number of labels per instance, and *label density* indicates the average number of labels per instance d relative to the total number of classes n.

dataset	n	#instances m	#attributes a	d	density
scene	6	2407	294 (86732)	1.07	17.9 %
emotions	6	593	72	1.87	31.1 %
yeast	14	2417	103 (10712)	4.24	30.3 %
tmc2007	22	28596	49060	2.16	9.8 %
genbase	27	662	1186	1.25	4.6 %
medical	45	978	1449	1.25	2.8 %
enron	53	1702	1001	3.39	6.4 %
mediamill	101	43907	120	4.38	4.3 %
rcv1-v2	101	804414	231188 (25000)	3.24	3.1 %
r21578	120	11367	21474 (10000)	1.26	1.0 %
bibtex	159	7395	1836	2.4	1.5 %
eurlex_sm	201	19348	166448 (5000)	2.21	1.1 %
eurlex_dc	410	19348	166448 (5000)	1.29	0.3 %
delicious	983	16105	500	19.02	1.9 %

5.2. Datasets

The datasets that were included in the experimental setup cover three application areas in which multilabeled data are frequently observed: *text categorization* (among others, the *Reuters-RCV1* and *Reuters-21578* datasets and the *EUR-Lex* dataset), *multimedia classification* (the *scene*, *mediamill* and *emotions* datasets) and *bioinformatics* (*yeast* and *genbase*). Table 2 provides an overview of the different characteristics of the used datasets.

The *Reuters Corpus Volume I (Reuters-RCV1)* is one of the most widely used test collection for text categorization research. It contains 804,414 newswire documents, which we split into 535,987 training documents (all documents before and including April 26th, 1999) and 268,427 test documents (all documents after April 26th, 1999). We used the token files of Lewis et al. (2004), which are already word-stemmed and stop word reduced. However we repeated the stop word reduction as we experienced that there were still a few occurrences. The 25,000 most frequent features on the training set were selected and weighted with TF-IDF weights (Salton and Buckley, 1988). We did not restrict the set of

The *Reuters-RCV1* dataset is available from `http://www.ai.mit.edu/projects/jmlr/papers/volume5/lewis04a/lyrl2004_rcv1v2_README.htm` (Lewis et al., 2004), the *Reuters-21578* dataset from `http://www.daviddlewis.com/resources/testcollections/reuters21578/`, *EUR-Lex* from `http://www.ke.tu-darmstadt.de/resources/eurlex/` and the remaining datasets from `http://mlkd.csd.auth.gr/multilabel.html`.

103 categories although one class does not contain any examples in the training set.

We also experimented with the older *Reuters-21578* corpus (Lewis, 1997), which has 11,367 examples and 120 possible labels. Through similar pre-processing as in the *Reuters-RCV1* dataset, we obtained 10,000 features for this dataset.

The *EUR-Lex* is a recent dataset containing 19,348 legislative documents from the European Union and is publicly available under `http://www.ke.informatik.tu-darmstadt.de/resources/eurlex/`. The documents are classified according to three different classification schemes: *subject matter* with 201 classes, *directory code* with 410 classes and *EUROVOC* with 3956 classes. However, we did not conduct experiments on the latter dataset since with almost 4000 classes we would need to maintain nearly 8 mio. perceptrons in memory. A special variant of MLPP was developed in order to be able to process datasets of this size (Loza Mencía and Fürnkranz, 2008a). After a similar pre-processing as for *RCV1* and *Reuters-21578*, we obtained 5,000 features.

Other text classification datasets include *medical* from a competition that aimed at assigning codes from the International Classification of Diseases to clinical free texts, the *enron* dataset of business-related emails from the Enron Corp. management, *bookmarks* and *bibtex*, collections from the social bookmarking platform BibSonomy, the *tmc2007* dataset of aviation safety reports assigned to flight problem types, and the large *delicious* dataset extracted from the *del.icio.us* social bookmarking platform. We used the pre-determined training/test splits.

The learning task in the *yeast* gene functional multiclass classification problem is to associate genes with a subset of 14 functional classes from the Comprehensive Yeast Genome Database of the Munich Information Center for Protein Sequences. Each of 2417 genes is represented with 103 features. In previous experiments (Loza Mencía and Fürnkranz, 2008b), we found that even the pairwise problems are hard to separate with a linear classifier (much more so in the binary relevance setting). Thus, in this set of experiments, we added all pairwise feature products to the original feature representation, in order to simulate a quadratic kernel function.

The task in the *scene* dataset (Boutell, Luo, Shen, and Brown, 2004) is to recognize which of six possible scenes (*beach, sunset, field, fall foliage, mountain, urban*) can be found in a 2407 pictures. Many pictures contain more than one scene. For each image, spatial color moments are used as features. Each picture is divided into 49 blocks using a 7×7 grid. A picture is then represented using the mean and the variance of each color band of each block, i.e., using a total of $2 \times 3 \times 7 \times 7 = 294$ features. Like in the *Yeast* dataset, we enriched the feature set with all pairwise feature products.

Furthermore, the *genbase* dataset contains a protein classification task. The dataset from the *mediamill* Challenge is dedicated to news video classification,

`http://mips.gsf.de/genre/proj/yeast/`

and in *emotions* the task is assign emotions to music.

5.3. Algorithmic Setup

All algorithms are trained incrementally. For the *RCV1* dataset, a single, chronological pass through the data was used (one epoch) because our previous results have shown that multiple iterations are not necessary (Loza Mencía and Fürnkranz, 2008b). For the remaining text classification we report the results for 10 epochs. The classifiers for the supposedly more difficult non-textual datasets were trained using 100 epochs. However, in terms of the relative order of the tested methods, we found that the results are quite insensitive to the exact numbers of epochs.

For *yeast*, *scene*, *Reuters-21578* and *EUR-Lex* the reported results are estimated from 10-fold cross-validation. In order to ensure that no information from the test set enters the training phase for the text datasets, the TF-IDF transformation and the feature selection were conducted only on the training sets of the cross-validation splits. For datasets for which it was not indicated we used the first two-thirds of examples for training and the remaining for testing. Specifically, we used 391 training examples for *emotions*, 21519 for *tmc2007*, 463 for *genbase*, 465 for *medical*, 1123 for *enron*, 30993 for *mediamill*, the aforementioned 535,987 for *rcv1-v2*, 4930 documents for *bibtex* and 12,920 for *delicious*.

All the perceptrons of the different algorithms were initialized with random values.

6. Evaluation

The following sections analyze, in short, the predictive quality and in a more extensive way the computational efficiency of the presented algorithms.

6.1. Computational Efficiency

Our analysis of computational efficiency concentrates on the savings in base classifier evaluations using the *QWeighted* method on the different multilabel datasets.

Table 3 depicts the gained reduction of prediction complexity of the *QWeighted* approach with respect to the classifier evaluations for CMLPP. For each of the four listed methods (BR, CMLPP, QCMLPP1 and QCMLPP2) the average number of base classifier evaluations is stated. In addition, for QCMLPP1 and 2 the ratio of classifier evaluations to the complete set of pairwise classifiers, which are typically evaluated in the CMLPP approach, are denoted within brackets, to emphasize the achieved reduction.

The first remarkable observation is the clear improvement when using the *QWeighted* approach. Except for the four smallest datasets regarding the labelsize, both variants of the QCMLPP use less than 20 percent of the classifier evaluations for CMLPP.

Another appreciable point, especially regarding the mentioned deviation, is the clearly visible correlation between the gained reduction and the label density of the problem, i.e. the ratio of the average number of labels per instance

Table 3: Computational costs at prediction in average number of classifier evaluations per instance. The italic values next to the two multilabel adaptations of *QWeighted* show the ratio of classifier evaluations to CMLPP and the second rightmost column describes the average number of relevant labels.

dataset	n	BR	CMLPP	QCMLPP1	QCMLPP2	$n \log(n)$	$n + dn \log(n)$	d	density $\frac{d}{n}$
scene	6	6	21	11.51 *(54.8%)*	11.46 *(54.6%)*	10.75	17.50	1.07	17.9 %
emotions	6	6	21	17.03 *(81.1%)*	16.59 *(79.0%)*	10.75	26.10	1.87	31.2 %
yeast	14	14	105	67.57 *(64.4%)*	64.99 *(61.9%)*	36.94	170.65	4.24	30.3 %
tmc2007	22	22	253	81.76 *(32.3%)*	78.01 *(30.8%)*	68.00	168.89	2.16	9.82 %
genbase	27	27	378	71.53 *(18.9%)*	62.11 *(16.4%)*	88.99	138.23	1.25	4.63 %
medical	45	45	1035	112.78 *(10.9%)*	103.67 *(10.0%)*	171.30	259.12	1.25	2.78 %
enron	53	53	1431	286.36 *(20.0%)*	262.30 *(18.3%)*	210.43	764.24	3.38	6.38 %
mediamill	101	101	5151	489.45 *(9.50%)*	378.04 *(7.34%)*	466.13	2142.64	4.38	4.34 %
rcv1-v2	103	103	5356	485.23 *(9.06%)*	456.23 *(8.52%)*	477.38	1649.70	3.24	3.15 %
r21578	120	120	7260	378.45 *(5.21%)*	325.94 *(4.49%)*	574.50	843.87	1.26	1.05 %
bibtex	159	159	12720	604.37 *(4.75%)*	492.73 *(3.87%)*	805.96	2093.29	2.40	1.51 %
eurlex_sm	201	201	20301	926.71 *(4.56%)*	771.62 *(3.80%)*	1065.96	2556.78	2.21	1.10 %
eurlex_dc	410	410	83845	1667.16 *(1.98%)*	1136.85 *(1.35%)*	2466.62	3591.95	1.29	0.31 %
delicious	983	983	483636	48680.12 *(10.1%)*	46835.89 *(9.68%)*	6773.47	129814.40	19.02	1.93 %

to the total number of labels. The dataset with the highest density, *emotions*, achieved the lowest reduction, followed by *yeast* with a similar density and reduction ratio. Similarly both QCMLPP variants evaluated the lowest ratio of classifiers for the dataset with the lowest density, the *eurlex_dc* dataset. This observation confirms the previously stated expectation that the reduction is highly influenced by the density. This effect is not surprising, since roughly speaking QCMLPP employs iteratively *QWeighted* until the calibrated label is found, and the number of iterations is obviously related to the density. Furthermore the results show that QCMLPP2 slightly but constantly outperforms QCMLPP1.

For estimating the average runtime in practice, two columns were included, which state the $n \log(n)$ and $n + dn \log(n)$ values for the corresponding datasets. We can clearly confirm that the number of classifier evaluations is for all considered datasets smaller than the previously estimated upper bound of $n + dn \log(n)$. Note that the value for *yeast* 170.65 is actually greater than the number of existing classifiers (105). This is due to the fact that the values lie yet in a range where lower order terms have still an impact in the equation.

Figure 12 visualizes the above results and allows again a comparison to different complexity values such as n, $n \log(n)$ and n^2. The upper figure is a recapitulation of the results from Park and Fürnkranz (2007) extended with multiclass classification performance results of the multilabel datasets considered in this paper: instead of evaluating until finding the calibrating label, *QWeighted* was only applied once such as if it was a multiclass problem. These results for the simulated multiclass classification performance support additionally the statement that *QWeighted* achieves an $n \log(n)$ runtime in practice. For better readability, a logarithmic scale for both axis is used. The lower figure is more interesting in this context, where multilabel classification prediction complexity of QCMLPP is presented. Note that the y-axis now describes the number of comparisons, respectively, classifier evaluations divided by the number of labels, which is graphically motivated and allows a finer distinction of the different curves. Note also that for the black curve $(n + dn \log(n))$, the actual average number of labels from data was used for computing the values and are identical to the ones from Table 3. These values are also depicted in the additional Figure 13, which shows again the comparison of computational costs split into two figures, the first for smaller datasets with $n < 103$ and the second for larger datasets. In comparison to Figure 12, the x-axis is now linear and we have added the dataset names to the data points.

As we can see from these figures, the empirical runtime bound $n + dn \log(n)$ is never exceeded. We conclude that this estimate is a reasonable indicator for the runtime complexity of QCMLPP.

6.2. Predictive Quality

Although it is not the focus of this study, we will compare in this section the prediction quality of BR and CMLPP in order to demonstrate the expected advantage of the pairwise approach. Note that the multilabel losses of the QCMLPP are exactly equal to those of CMLPP since both compute for every

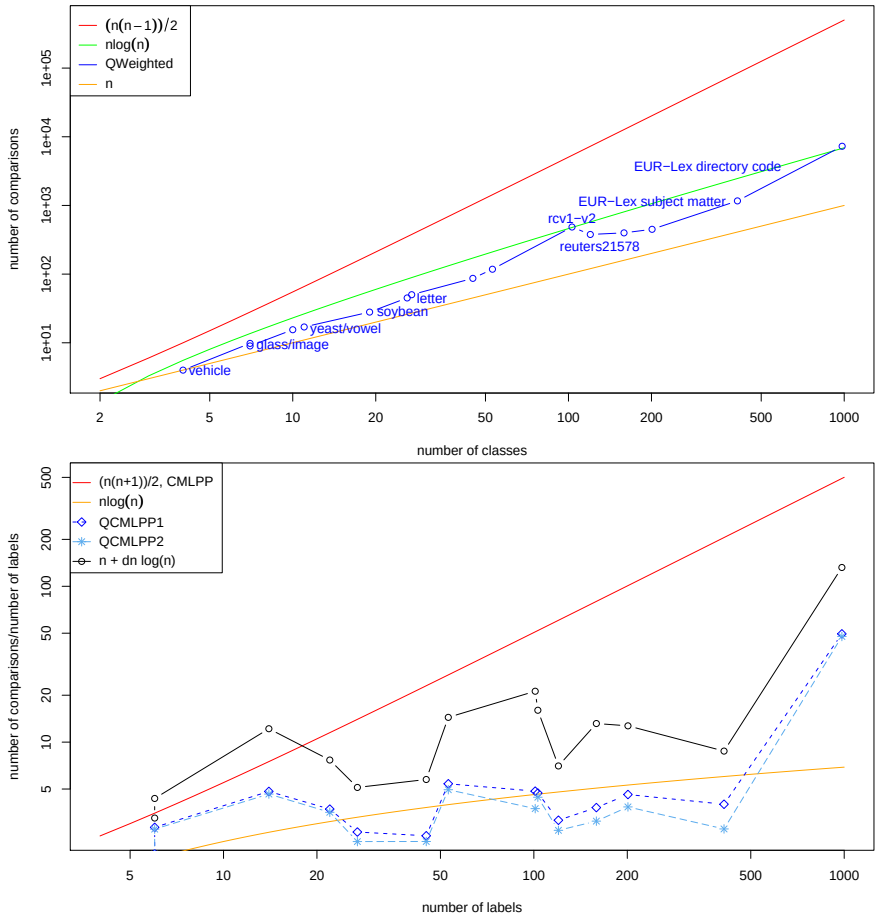


Figure 12: Prediction complexity of *QWeighted* and QCMLPP: number of comparisons needed in dependancy of the number of classes n for different multiclass and multilabel problems. *Upper figure*: Problems *vehicle* to *letter* in the first figure are multiclass problems already analyzed by Park and Fürnkranz (2007), while multiclass versions of the multilabel datasets as described in Table 2 were evaluated within this study. *Lower figure*: QCMLPP1/2 is compared to $n(n+1)/2$ as in CMLPP, n as in BR and $n \log(n)$ on 14 multilabel datasets.

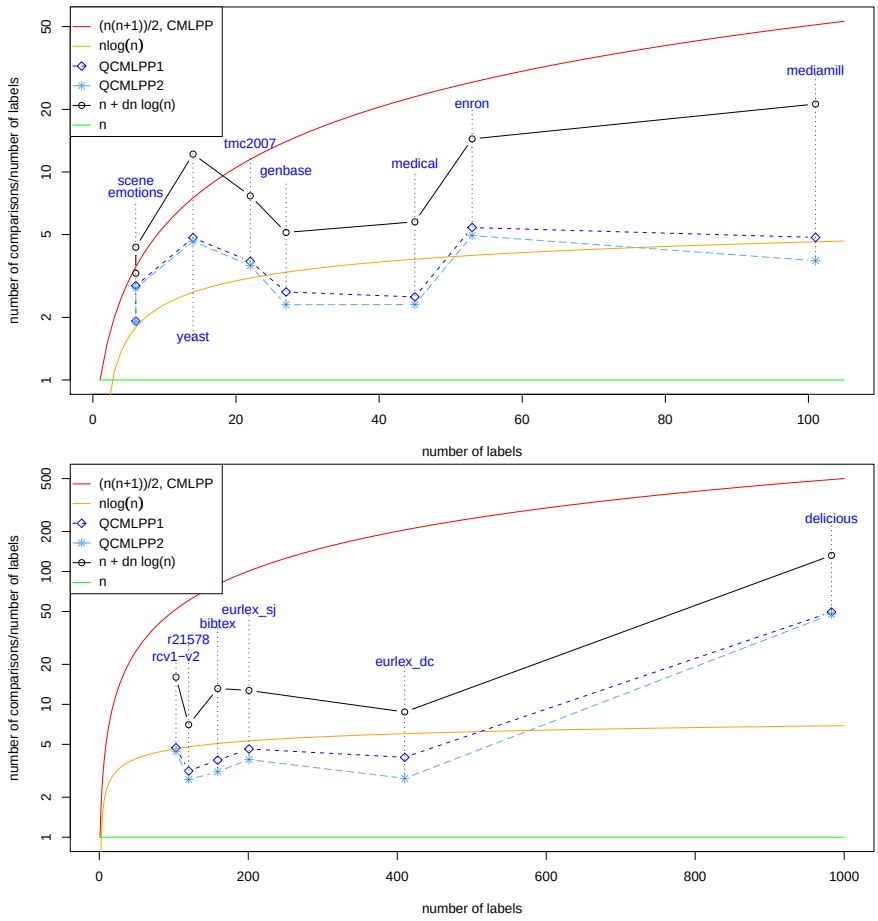


Figure 13: Prediction complexity of QCMLPP : the upper figure contains the small datasets, the bottom figure the large datasets

instance the same partitioning into relevant and irrelevant labels. Table 4 shows the label set predictions performance according to Section 5.1.

The first remarkable observation is that for the overall evaluation measures HamLoss and F1 the pairwise approach dominates the one-per-class approach for every dataset except *genbase* and *medical*. BR's Prec is even outperformed for these datasets. On the other hand, QCMLPP achieves a lower Rec for the datasets with slightly more than 100 classes, beginning at *reuters-21578* with 120 classes. This is due to the fact that the calibration tends to underestimate the number of returned labels for each instance, especially for a high number of total classes. A possible explanation for this behavior is the following: when the binary relevance classifiers, that are also included in CMLPP, predict that v classes are positive, then this means for the remaining classes that they have to

Table 4: Multilabel performance of the different algorithms. For HamLoss low values are good, for the other three measures the higher the better. Bold values represent the best value for each dataset and measure combination. Note that the multilabel losses of QCMLPP are exactly equal to those of CMLPP.

		HamLoss		Precision		Recall		F1	
dataset	n	BR	CMLPP	BR	CMLPP	BR	CMLPP	BR	CMLPP
scene	6	10.42	**10.00**	71.80	**71.83**	71.21	**74.20**	71.19	**72.76**
emotions	6	35.64	**34.08**	46.78	**48.62**	60.15	**61.90**	52.63	**54.47**
yeast	14	24.09	**22.67**	60.47	**62.37**	59.07	**63.31**	59.76	**62.83**
tmc2007	22	7.37	**6.78**	62.57	**64.16**	66.47	**73.61**	64.46	**68.56**
genbase	27	**0.26**	0.48	99.22	**99.59**	**95.49**	90.60	**97.32**	94.88
medical	45	**1.51**	1.51	71.72	**76.02**	**75.84**	66.75	**73.72**	71.08
enron	53	7.56	**6.01**	41.56	**52.82**	47.05	**49.51**	44.13	**51.11**
mediamill	101	4.52	**4.16**	42.28	**56.66**	10.05	**19.70**	16.24	**29.23**
rcv1-v2	103	1.26	**1.03**	80.15	**84.89**	79.70	**81.61**	79.93	**83.22**
r21578	120	0.78	**0.55**	59.98	**72.89**	**78.36**	76.68	67.92	**74.63**
bibtex	159	1.57	**1.35**	46.53	**57.97**	**36.30**	34.84	40.78	**43.53**
eurlex_sm	201	0.76	**0.54**	63.39	**77.88**	**74.11**	71.57	68.32	**74.59**
eurlex_dc	410	0.26	**0.17**	56.26	**79.21**	**70.54**	61.98	62.58	**69.54**
delicious	983	5.58	**3.48**	11.88	**19.77**	**29.59**	26.51	16.95	**22.65**

obtain at least $n - v$ votes of their maximum of n votes in order to be predicted as positive. The probability that this happens for a real positive class decreases with increasing n, since it becomes more probable that at least v base classifiers mistakenly take a wrong decision. However, a look at the avg. predicted label set size shows that this is only the case for the *EUR-Lex* datasets and not for *Reuters-21578* or *delicious*. For *delicious* QCMLPP even predicts more than 25 instead of 19 labels. On the other hand we can observe that BR always predicted a higher label number than QCMLPP on the dataset where it achieved a higher Rec. One extreme are the 47 predicted labels for *delicious*, but note that in general it cannot be stated that BR overestimates the number of labels.

Note that it is easily possible to bias the recall/precision trade-off of the calibration by simply subtracting or adding a fixed number of votes to the artificial class count.

6.3. Support Vector Machines

Such as BR, MLPP is potentially able to use any binary classifier as base classifier. Therefore, we conducted experiments with support vector machines as base learners in order to demonstrate that the same positive effects can also be expected from the pairwise approach and the *QWeighted* optimization when using a different base learner. We used the LIBSVM implementation (Chang and Lin, 2001) with standard settings for the non-textual datasets and the efficient LIBLINEAR implementation (Fan et al., 2008) for textual datasets with the primal L2-loss SVM option, which is supposed to enhance speed (Hsu, Chang,

Table 5: SVM as base learner - Computational costs at prediction in average number of classifier evaluations per instance. The italic values next to the multilabel adaptation of *QWeighted* (QCMLPP2) shows the ratio of classifier evaluations to CMLPP and the second rightmost column describes the average number of relevant labels.

dataset	n	BR	CMLPP	QCMLPP2	$n \log(n)$	$n + dn \log(n)$	d
scene	6	6	21	7.88 *(37.5%)*	10.75	17.50	1.07
emotions	6	6	21	11.87 *(56.5%)*	10.75	26.10	1.87
yeast	14	14	105	40.31 *(38.4%)*	36.94	170.65	4.24
tmc2007	22	22	253	68.92 *(27.2%)*	68.00	168.89	2.16
medical	45	45	1035	97.40 *(9.41%)*	171.30	259.12	1.25
enron	53	53	1431	223.42 *(15.6%)*	210.43	764.24	3.38
r21578	120	120	7260	303.90 *(4.19%)*	574.50	843.87	1.26
bibtex	159	159	12720	485.97 *(3.82%)*	805.96	2093.29	2.40

and Lin, 2009). We ignored the results on *genbase* since LIBSVM predicted the empty label set on all test examples. For the remaining missing datasets no results could be retrieved due to the higher memory requirements of the SVMs compared to the simple perceptrons. For *yeast* and *scene* we did not use the quadratic kernel simulation.

Table 5 shows the computational costs of QCMLPP2 with SVM as base classifier. We can observe an overall similar picture compared to the results of Table 3, the pairwise approach clearly benefits from the *QWeighted* optimization. However, while the reduction in number of required comparisons for the textual datasets is very similar, using LIBSVM seems to allow to further improve the ratio on the non-textual *scene*, *emotions* and *yeast*. The explanation can be seen in Table 6, which lists the prediction quality for BR and QCMLPP2: A very high precision is achieved by LIBSVM for these datasets due to predicting only a small number of labels. This cautious behavior of LIBSVM could already be observed for the *genbase* dataset. QCMLPP2 with perceptrons as base classifier e.g. predicts 2.51 labels in average on the *emotions* test set, while with SVM as base classifier only 1.27 are predicted. This means for QCMLPP2 in average more than one additional *QWeighted* iteration for each example during classification, which is the reason for the further reduction of the computational costs.

Note that although the obtained reductions in number of base classifiers is similar for both perceptrons and SVM, training the SVMs does usually require a higher amount of CPU-time. Except for *emotions*, for which the time is almost equal, and *yeast* and *scene*, which are not directly comparable due to the different feature representations used, the perceptrons are always faster, namely 2.3× faster for *tmc2007* to even 29 × faster for *enron*.

Especially if we consider that the prediction quality of perceptrons and SVMs are very similar (at least for the text classification tasks), this constitutes an important point in defense of the perceptron algorithm. However, it is also interesting to observe that the distance between BR and QCMLPP is consid-

Table 6: Multilabel performance of the different algorithms with SVM as base learner. For HamLoss low values are good, for the other three measures the higher the better. Bold values represent the best value for each dataset and measure combination. Note that the multilabel losses of QCMLPP are exactly equal to those of CMLPP.

		HamLoss		Precision		Recall		F1	
dataset	n	BR	CMLPP	BR	CMLPP	BR	CMLPP	BR	CMLPP
scene	6	12.57	**12.51**	**93.25**	93.04	32.16	**32.58**	47.77	**48.21**
emotions	6	27.56	**26.57**	**65.55**	64.98	34.34	**41.85**	45.07	**50.91**
yeast	14	22.51	22.51	**75.61**	75.60	37.81	**37.82**	50.41	**50.41**
tmc2007	22	6.99	**6.63**	66.16	**67.31**	62.33	**66.16**	64.19	**66.73**
medical	45	**1.09**	1.11	**83.12**	82.10	76.56	**76.79**	**79.70**	79.36
enron	53	5.70	**5.22**	55.87	**59.95**	48.64	**53.36**	52.00	**56.47**
r21578	120	0.56	**0.55**	71.23	**71.76**	**78.49**	78.34	74.68	**74.90**
bibtex	159	1.48	**1.39**	50.45	**54.65**	37.60	**39.32**	43.09	**45.73**

erably reduced when using SVMs, which might be an indication for a higher robustness against weak base classifier for the pairwise approach.

7. Conclusions

Multilabel classification is becoming a more and more important task in machine learning due to the increasing amount of application scenarios where it is necessary to not only predict one top class as in multiclass classification, but a set of relevant classes. The common approach of training one classifier for each class that determines a binary relevance is clearly outperformed by the approach of learning pairwise preferences between pairs of classes. The main disadvantage of this approach was, until now, the quadratic number of base classifiers needed and hence the increased computational costs for prediction and the increased memory requirements. We have presented in this paper a time efficient efficient algorithm based on the pairwise approach.

The proposed approach combines a technique that transforms a class ranking into a bipartite prediction by introducing an artificial thresholding class, called calibration (Fürnkranz et al., 2008), with the *QWeighted* voting that stops the computation of the ranking when the bipartite separation is already determined (Park and Fürnkranz, 2007). For the combined *QWeighted* multilabel method the computational costs savings compared to the normal voting are especially important with increasing number of classes. Though not analytically proven, our empirical results show that the complexity is upper bounded by $n + dn \log(n)$, in comparison to the evaluation of n in the case of the one-per-class approach and $O(n^2)$ for the unmodified pairwise approach. For the *QWeighted* multilabel approach, we see improvements in a more appropriate integration of the *QWeighted* concept, namely to identify and exploit unnecessary classifier evaluations to the multilabel setting. In this context, QCMLPP2 was already a step forward.

The benefit in predictive quality of using CMLPP against using BR was shown by an extensive experimental evaluation on 14 datasets. Together with *QWeighted* CMLPP is able to achieve a good trade-off between predictive quality and speed in the multilabel setting. Additional experiments using state-of-the-art support vector machines as base learner instead of the perceptron algorithm initially used in MLPP confirmed that the binary relevance approach is outperformed by the pairwise approach. These experiments also show that the advantage of using the pairwise approach and *QWeighted* is independent of the base learner employed.

The key remaining bottleneck is that we still need to store a quadratic number of base classifiers, because each of them may be relevant for some example. We are currently investigating to combine QCMLPP with HOMER, an algorithm that arranges multilabel base classifiers in a hierarchical tree in order to reduce computational costs and memory consumption an setting with a high number of classes (Tsoumakas et al., 2008). First experimental results are very encouraging (Tsoumakas et al., 2009).

Acknowledgements

This work was supported by the EC 6^{th} Framework project *ALIS (Automated Legal Information System)* and by the *German Science Foundation (DFG)*.

References

Angulo, C., Ruiz, F., González, L., Ortega, J. A., 2006. Multi-classification by using tri-class svm. Neural Processing Letters 23 (1), 89–101.

Bishop, C. M., 1995. Neural Networks for Pattern Recognition. Oxford University Press.

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

Brinker, K., Fürnkranz, J., Hüllermeier, E., 2006. A Unified Model for Multilabel Classification and Ranking. In: Brewka, G., Coradeschi, S., Perini, A., Traverso, P. (Eds.), Proceedings of the 17th European Conference on Artificial Intelligence (ECAI-06).

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

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., Singer, Y., Mar. 2006. Online passive-aggressive algorithms. Journal of Machine Learning Research 7, 551–585.

Crammer, K., Singer, Y., 2003. A Family of Additive Online Algorithms for Category Ranking. Journal of Machine Learning Research 3 (6), 1025–1058.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., Lin, C.-J., aug 2008. Liblinear: A library for large linear classification. Journal of Machine Learning Research 9, 1871–1874.

Freund, Y., Schapire, R. E., 1999. Large Margin Classification using the Perceptron Algorithm. Machine Learning 37 (3), 277–296.

Fürnkranz, J., 2002. Round Robin Classification. Journal of Machine Learning Research 2, 721–747.

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

Hsu, C.-W., Chang, C.-C., Lin, C.-J., May 2009. A practical guide to support vector classification. Tech. rep., Department of Computer Science, National Taiwan University, available at `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`.

Hsu, C.-W., Lin, C.-J., 2002. A Comparison of Methods for Multi-class Support Vector Machines. IEEE Transactions on Neural Networks 13 (2), 415–425.

Katakis, I., Tsoumakas, G., Vlahavas, I., 2008. Multilabel text classification for automated tag suggestion. In: Proceedings of the ECML/PKDD-08 Workshop on Discovery Challenge. Antwerp, Belgium.

Khardon, R., Wachman, G., Feb. 2007. Noise tolerant variants of the perceptron algorithm. Journal of Machine Learning Research 8, 227–248.

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

Lewis, D. D., Yang, Y., Rose, T. G., Li, F., 2004. Rcv1: A new benchmark collection for text categorization research. Journal of Machine Learning Research 5, 361–397.

Li, Y., Zaragoza, H., Herbrich, R., Shawe-Taylor, J., Kandola, J. S., 2002. The Perceptron Algorithm with Uneven Margins. In: Machine Learning, Proceedings of the Nineteenth International Conference (ICML 2002). pp. 379–386.

Loza Mencía, E., Fürnkranz, J., 2008a. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In: Daelemans, W., Goethals, B., Morik, K. (Eds.), Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Disocvery in Databases (ECML-PKDD-2008), Part II. Springer-Verlag, Antwerp, Belgium, pp. 50–65.

Loza Mencía, E., Fürnkranz, J., 2008b. Pairwise learning of multilabel classifications with perceptrons. In: Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IJCNN 08). Hong Kong, pp. 2900–2907.

Loza Mencía, E., Park, S.-H., Fürnkranz, J., 2008. Advances in efficient pairwise multilabel classification. Tech. Rep. TUD-KE-2008-06, TU Darmstadt, Knowledge Engineering Group, available at `http://www.ke.informatik.tu-darmstadt.de/publications/reports/tud-ke-2008-06.pdf`.

Moreira, M., Mayoraz, E., 1998. Improved pairwise coupling classification with correcting classifiers. In: Nedellec, C., Rouveirol, C. (Eds.), ECML. Vol. 1398 of Lecture Notes in Computer Science. Springer, pp. 160–171.

Park, S.-H., Fürnkranz, J., 2007. Efficient pairwise classification. In: Kok, J. N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (Eds.), Proceedings of 18th European Conference on Machine Learning (ECML-07). Springer-Verlag, Warsaw, Poland, pp. 658–665.

Rosenblatt, F., 1958. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological Review 65 (6), 386–408.

Salton, G., Buckley, C., 1988. Term-weighting approaches in automatic text retrieval. Inf. Process. Manage. 24 (5), 513–523.

Shalev-Shwartz, S., Singer, Y., 2005. A New Perspective on an Old Perceptron Algorithm. In: Learning Theory, 18th Annual Conference on Learning Theory (COLT 2005). Springer, pp. 264–278.

Tsampouka, P., Shawe-Taylor, J., 2007. Approximate maximum margin algorithms with rules controlled by the number of mistakes. In: Ghahramani, Z. (Ed.), Machine Learning, Proceedings of the Twenty-Fourth International Conference on Machine Learning(ICML 2007). Vol. 227 of ACM International Conference Proceeding Series. ACM, pp. 903–910.

Tsoumakas, G., Katakis, I., Loza Mencía, E., Park, S.-H., Fürnkranz, J., 2009. A comparison of decompositive multi-label classification models. In: Proceedings of ECML/PKDD 2009 Workshop on Preference Learning.

Tsoumakas, G., Katakis, I., Vlahavas, I., 2008. Effective and efficient multilabel classification in domains with large number of labels. In: Proceedings of the ECML/PKDD-08 Workshop on Mining Multidimensional Data (MMD-08). Antwerp, Belgium.