Incorporating Prior Knowledge into Boosting for Multi-Label Classification

Xiao Wang
The Key Laboratory of Embedded System and Service Computing
Ministry of Education,
Department of Control Science and Engineering,
Tongji University,
Shanghai 201804, China
E-mail: pandaxiaoxi@gmail.com

Guo-Zheng Li*
The Key Laboratory of Embedded System and Service Computing
Ministry of Education,
Department of Control Science and Engineering,
Tongji University,
Shanghai 201804, China
E-mail: gzli@tongji.edu.cn
*Corresponding author

Abstract: Multi-label learning deals with the problem where each instance may belong to multiple labels simultaneously. The task of the learning paradigm is to output the label set whose size is unknown a priori for each unseen instance, through analyzing the training data set with known label sets. Existing multi-label learning algorithms are almost based on the purely data-driven method. The larger the training dataset, the better the performance of the classifier. However, in some cases, training dataset is too small to obtain an accurate model, while there are some prior knowledge available. In this paper, a novel boosting based multi-label learning algorithm called KnowBoost.MH is proposed. It is derived from the famous AdaBoost.MH algorithm by incorporating prior knowledge into boosting to compensate for the lack of training data. Experimental results on two real-world multi-label datasets show that KnowBoost.MH outperforms AdaBoost.MH and some of well-established multi-label learning algorithms especially in the case of the lack of training data.

Keywords: Multi-Label Learning, Boosting, Prior Knowledge.

Biographical notes: Xiao Wang received his B.Sc. and M.Sc. degree in Computer Science at Henan Normal University and University of Electronic Science and Technology of China, China, in 2004 and 2007 respectively. He is currently a Ph.D. candidate in Department of Control Science and Engineering, Tongji University, Shanghai 201804, China.
1 Introduction

Multi-label learning studies how to analyze data sets with a set of labels in each instance, which has been one of the widely studied machine learning frameworks. Multi-label learning originated from the problems in the text categorization field and covers many real-world learning problems. For instance, in text categorization, some predefined topics may be assigned to each document, such as government, sports and health (McCallum, 1999; Schapire & Singer, 2000); in bioinformatics, each gene may belong to several functional classes, such as metabolism, transcription and protein synthesis (Elisseeff & Weston, 2002); in scene classification, each scene image may be associated with a set of semantic classes, such as beach and urban (Boutell et al., 2004). In all these cases, each instance in the training set belongs to a set of labels and the task is to predict the label set whose size is unknown a priori for each unseen instance.

A lot of previous works have only been focused on binary or multi-class problems and dimension reduction (You et al., 2011; Liang Lan, 2011; Zeng et al., 2009). Traditional binary or multi-class problems may be regarded as the special cases of multi-label ones, where the generality inevitably makes it more difficult to learn from such multi-label data. Most previous studies of multi-label learning can be categorized into two types (Grigorios Tsoumakas, 2007): (1) problem transformation, and (2) algorithm adaptation. Recently, many multi-label learning algorithms have been proposed. In problem transformation, typical ones include Label-powerset (LP) (Tsoumakas et al., 2010), pruned problem transformation (PPT) (Read, 2008), Binary relevance (BR), Ranking by pairwise comparison (RPC) (Eyke Hllermeier & Brinker, 2008), and Calibrated label ranking (CLR) (Frankranz et al., 2008), etc. On the other hand, typical ones of algorithm adaptation include extending the famous AdaBoost learning algorithm (Freund & Schapire, 1997) for multi-label data (Schapire & Singer, 1999, 2000); using decision trees to produce multi-label rules that can be understood by humans (Clare & King, 2001; De Comit et al., 2003); using different probabilistic generative models based text frequencies for multi-label text classification (McCallum, 1999; Ueda & Saito, 2003; Streich & Buhmann, 2008); using kernel functions for
multi-label problems (Boutell et al., 2004; Elisseeff & Weston, 2002; Godbole & Sarawagi, 2004; Kazawa et al., 2005); utilizing the popular instance-based lazy methods to predict labels of each unseen instance based on its similarity with training instances (Brinker et al., 2006; Zhang & Zhou, 2007; Younes et al., 2009); and adapting neural networks to multi-label setting (Zhang & Zhou, 2006; Zhang, 2009), etc. Recently, several algorithms have been proposed to improve the performance of classifiers by utilizing extra information provided by unlabeled data (Yi Liu & Yang, 2006; Chen et al., 2008; Li et al., 2010).

The existing multi-label learning algorithms are purely data-driven. A larger training dataset is required for better performance of the classifier. However, in some cases, training dataset is too small to obtain an accurate model, while there are some prior knowledge available. In this paper, a novel algorithm of knowledge based multi-label boosting called KnowBoost.MH is proposed, which is derived from the famous AdaBoost.MH algorithm (Schapire & Singer, 1999, 2000), through incorporating prior knowledge into boosting to compensate for the lack of the training data. Our proposed algorithm allows to incorporate prior knowledge of any form, as long as it can be used to estimate conditional probabilities of labels for each instance. In this paper, we use two benchmark text categorization datasets because keywords based prior knowledge is easily obtained. Experimental results on two real-world multi-label data sets show that KnowBoost.MH outperforms AdaBoost.MH and some of well-established multi-label learning algorithms especially in the case of the lack of training data.

The rest of this paper is arranged as follows. Section 2 gives a brief description of AdaBoost.MH, and then our proposed multi-label learning algorithm named KnowBoost.MH is introduced. Section 3 presents experimental datasets, settings, and prior models. Experimental results and discussions are shown in Section 4. Finally, Section 5 comes to the conclusion.

2 Prior Knowledge based Multi-label Boosting

2.1 A Brief Description of AdaBoost.MH

Let \( \chi \) denote the domain of instances and let \( \mathcal{Y} \) be a finite set of \( k \) possible labels. Given a multi-label training set \( \mathcal{T}_r = \{(x_1, Y_1), (x_2, Y_2), \ldots, (x_m, Y_m)\} \), where \( x_i \in \chi \) is a single instance and \( Y_i \subseteq \mathcal{Y} \) is the set of labels to each of which \( x_i \) belongs. The goal of the learning system is to output a multi-label classifier \( h : \chi \rightarrow 2^\mathcal{Y} \) which can predict the label set for each test instance. But instead of outputting the multi-label classifier, the learning system will output a function of the form \( f : \chi \times \mathcal{Y} \rightarrow \mathbb{R} \). For a given instance \( x \), the labels in \( \mathcal{Y} \) should be ranked according to their values of \( f(x, \cdot) \). A label \( l_1 \) is considered to be ranked higher than \( l_2 \) if \( f(x, l_1) \geq f(x, l_2) \). If \( \mathcal{Y} \) is the associated set of labels for \( x \), then a successful system will tend to rank labels in \( \mathcal{Y} \) higher than those not in \( \mathcal{Y} \).

AdaBoost.MH (see Algorithm 1) is one of the family of boosting algorithms, which generates a highly accurate classifier (also called final hypothesis) by combining a set of moderately accurate classifiers (also call weak hypothesis).
Algorithm 1: AdaBoost.MH

Input: A training set \( T_r = \{(x_1, Y_1), (x_2, Y_2), \ldots, (x_m, Y_m)\} \)
where \( x_i \in \chi, Y_i \subseteq \mathcal{Y} \) for \( i = 1, \ldots, m \)

Output: The final hypothesis \( f(x, l) = \sum_{t=1}^{T} h_t(x, l) \)

Process:
1. Initialize \( D_1(i, l) = 1/(mk) \) for \( i = 1, \ldots, m \) and \( l = 1, \ldots, k \);
2. for \( t = 1, \ldots, T \) do
3. Pass distribution \( D_t \) to the weak learner;
4. Get the weak hypothesis \( h_t : \chi \times \mathcal{Y} \rightarrow R \) from the weak learner;
5. Update:
6. \[ D_{t+1}(i, l) = \frac{D_t(i, l) \exp(-Y_i[l] h_t(x_i, l))}{Z_t} \]
7. where \( Z_t = \sum_{i=1}^{m} \sum_{l=1}^{k} D_t(i, l) \exp(-Y_i[l] h_t(x_i, l)) \) is a
   normalization factor chosen so that \( D_{t+1} \) will be a distribution
8. end

AdaBoost.MH mainly maintains a set of weights as a distribution \( D_t \) over
instances and labels. Initially, the distribution \( D_1 \) is uniform. On each round \( t \),
the distribution \( D_t \) (together with the training set \( T_r \)) is passed to the weak
learner who generates a weak hypothesis \( h_t \). The weak hypothesis is a function of
the form \( h_t : \chi \times \mathcal{Y} \rightarrow R \). The sign of \( h_t(x, l) \) is interpreted as a prediction on whether
the instance \( x \) belongs to the label \( l \), i.e., \( h_t(x, l) > 0 \) shows that \( x \) is considered
to belong to \( l \) while \( h_t(x, l) < 0 \) shows that it is considered not to belong to \( l \).
The absolute value of \( h_t(x, l) \) is interpreted as the strength of the belief in the
prediction.

AdaBoost.MH is derived by using a natural reduction of multi-label data to
binary data. Under the reduction, each instance \( (x, Y) \) is mapped to \( k \) binary-
labeled instances of the form \( ((x, l), Y[l]) \) for all \( l \in \mathcal{Y} \). In other words, we can think
of each label set \( Y \) as specifying \( k \) binary labels (depending on whether a label \( l \)
is included in \( Y \) or not), and then we can apply binary AdaBoost to the derived
binary data.

On each round \( t \), all the weights \( D_t(i, l) \) are updated to \( D_{t+1}(i, l) \) in a manner
as described below:

\[ D_{t+1}(i, l) = \frac{D_t(i, l) \exp(-Y_i[l] h_t(x_i, l))}{Z_t} \tag{1} \]

where \( Y_i[l] \) is defined to be 1 if \( l \in Y_i \) and -1 otherwise, and

\[ Z_t = \sum_{i=1}^{m} \sum_{l=1}^{k} D_t(i, l) \exp(-Y_i[l] h_t(x_i, l)) \tag{2} \]

is a normalization factor chosen so that \( D_{t+1} \) will be a distribution, i.e.,
\[ \sum_{i=1}^{m} \sum_{l=1}^{k} D_{t+1}(i, l) = 1 \]
Equation (1) means that if a example-label pair \( (x_i, l) \) is misclassified by \( h_t \)(i.e., \( Y_i[l] \) and \( h_t(x_i, l) \) differ in sign), its new weight \( D_{t+1}(i, l) \)
is increased; likewise, if a example-label pair \((x_i, l)\) is correctly classified by \(h_t\) (i.e., \(Y_i[l] = h_t(x_i, l)\) have same sign), its new weight \(D_{t+1}(i, l)\) is decreased. More detailed description may be found in Schapire & Singer (1999, 2000).

2.2 KnowBoost.MH

Here, we describe our proposed knowledge-based version of AdaBoost.MH, named KnowBoost.MH, that is explicitly designed to work on the training set and prior knowledge, and is able to compensate for the lack of the training data. We mainly deal with the problem that given a small number of labeled training data and prior knowledge, the task is to train a good classifier that minimizes the prediction error on an unseen instance.

The pseudo-code description of KnowBoost.MH is given in Algorithm 2. \(T_r\) is the training set as shown in Subsection 2.1 and \(T_e\) is the test set. \(T_p\) is the pseudo dataset generated by applying the prior knowledge into the test set. The weak learner is trained on the combination of the training set \(T_r\) with the pseudo dataset \(T_p\). In the pseudo dataset \(T_p\), besides the instances \((x_1, x_2, \ldots, x_n)\) and their associated set of labels \((Y_1, Y_2, \ldots, Y_n)\), there are their corresponding set of confidence values \((C_1, C_2, \ldots, C_n)\). That is, each \(c_j \in C_i\), where \(c_j \in (0, 1]\), shows the confidence level of \(y_j \in Y_i\) labeling. Given an instance \(x_i\) and its associated label set \(Y_i \subseteq Y\). Let us denote \(C_i\) as the confidence vector for \(x_i\), where its \(l\)-th component \(C_i[l]\) \((l \in Y)\) takes the confidence value of \(x\) belonging to label \(l\).

Intuitively, we directly train the prediction model on the combination of the pseudo dataset and the training set by employing the AdaBoost.MH algorithm. Through conducting the corresponding experiments, however, we observed that its effectiveness even be worse than that of not employing the pseudo dataset. The reason is that the prior knowledge is rough and generate the pseudo dataset with low equality, which will introduce noise. Instances with lower confidence are potential noisy source. A threshold \(\theta\) can distinguish instances with lower confidence from ones with higher confidence. We call instances with confidence value less than \(\theta\) as instances with lower confidence, otherwise, instances with higher confidence. For any instances with lower confidence, when they are wrongly predicted by the learned model, they could be noisy data generated by the rough prior knowledge. Thus in our proposed approach, we decrease the weights of these instances in order to weaken their effects. For instances with higher confidence, their weights are adjusted just like the original training instances. As can be seen from the algorithm, on each round, if a pseudo training instance with lower confidence is wrongly predicted, the instance may be noisy data and likely conflict with the learned model. Therefore, we decrease its training weight to weaken its effect by multiplying its weight by \(\beta||h(x, l) - Y_i[l]||\), where \(\beta||h(x, l) - Y_i[l]|| \in (0, 1]\). In the next round, the misclassified pseudo training instance with lower confidence will affect the further learning process less than the current round. After several rounds, the misclassified pseudo training instances with lower confidence will have lower weights, while the pseudo training instances with lower confidence which is correctly predicted will get larger weights. All of instances with large training weights will help the learning algorithm to train better classifiers.
Algorithm 2: KnowBoost.MH

**Input:** A training set $Tr = \{(x_1, Y_1), \ldots, (x_m, Y_m)\}$, a pseudo dataset $Tp = \{(x_1, Y_1, C_1), \ldots, (x_n, Y_n, C_n)\}$ and the threshold $\theta$

**Output:** The final hypothesis $f(x, l) = \sum_{t=1}^{T} h_t(x, l)$

**Process:**
1. Initialize $D_1(i, l) = W_1(i, l) = 1 = (\frac{m+n}{k})$ for $i = 1, \ldots, m+n$ and $l = 1, \ldots, k$;
2. for $t = 1, \ldots, T$ do
   3. Pass distribution $D_t$ over $Tr \cup Tp$ to the weak learner;
   4. Get the weak hypothesis $h_t: \chi \times \mathcal{Y} \rightarrow R$ from the weak learner;
   5. Set $\beta$:
      $\beta = \frac{1}{1 + \sqrt{2 \ln n/T}}$;
   6. Update:
      $$W_{t+1}(i, l) = \begin{cases} W_t(i, l)^{\beta_{h_t(x, l), Y_i[l]}C_i[l]} < \theta \\ W_t(i, l) \exp\left(-Y_i[l] h_t(x, l)\right)C_i[l] \geq \theta \end{cases}$$
   7. Normalize the weights to a distribution:
      $$D_{t+1}(i, l) = W_{t+1}(i, l) / \sum_{i=1}^{m+n} \sum_{l=1}^{k} W_{t+1}(i, l)$$
3. Experimental Data Sets and Settings

3.1 Data Sets

In this paper, two real-world data sets are used to evaluate the performance of each compared algorithm, whose statistics are summarized in Table 1 below:

- **Reuters-21578 dataset**
  The Reuters collection is the most commonly-used collection for Text Categorization evaluation. The Reuters-21578 Distribution 1.0 is used in this paper. All documents without any topic label or with empty main text are discarded from the collection. Each remaining document belongs to at least one of the 135 possible topics. The 10 most frequent topics are used for experiments. For each remaining document, the following preprocessing operations are performed prior to experiments: All words were converted to lower case, punctuation marks and strings of digits were removed, and “function words” from the SMART stop-list were removed. The simple term selection method based on document frequency is used to reduce the dimensionality of the data set. Actually, only about 2% words with highest document frequency are retained in the final vocabulary. Each document in the data set is described as a feature vector using the "Set-of-Words" representation (Dumais et al., 1998), i.e. each dimension of the feature vector
are represented by only binary feature values corresponding to a word either appearing or not in a document. Finally, each document is represented by a 478-dimensional feature vector.

- **20 Newsgroups dataset**
  The 20 Newsgroups data set is a collection of approximately 20,000 newsgroup articles, partitioned (nearly) evenly across 20 different newsgroups (Lang, 1995). The 20 newsgroups collection has become a popular data set for experiments in applications of machine learning techniques, such as text classification and text clustering. This data set was originally treated as single-label one, as reported in Joachims (1997). However, since people tend to post articles to multiple newsgroups, we found that about 4.5% of the articles are actually multi-labeled after checking the "Newsgroups" header of the articles. The 20 different newsgroups are used as labels for experiments. The above mentioned processing operations are performed for the data set prior to experiments. Finally, each article is represented by a 2391-dimensional feature vector.

<table>
<thead>
<tr>
<th>Dataset</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S</td>
<td>dim(S)</td>
<td>L(S)</td>
<td>DL(S)</td>
<td>LCard(S)</td>
</tr>
<tr>
<td>Reuters-21578</td>
<td>7907</td>
<td>478</td>
<td>10</td>
<td>44</td>
<td>1.118</td>
</tr>
<tr>
<td>20 Newsgroups</td>
<td>19335</td>
<td>2391</td>
<td>20</td>
<td>54</td>
<td>1.045</td>
</tr>
</tbody>
</table>

### 3.2 Prior Knowledge

Our proposed method permits prior knowledge of any form, as long as it provides estimates, however rough, of the probability of any instance belonging to any label. Here we describe one similar technique as used in Wu & Srihari (2004) for creating such a rough prior model. Ideally, one can come up with a list of keywords for each label. These keywords are produced through a rather subjective process based on only the general understanding of what the labels are about. These keywords used by two datasets are listed in Table 2 and 3, respectively.

We use these keywords to build a rather simple model to predict the probability of an instance belonging to a label. We purposely use a model that is far from perfect to see how our proposed method performs with prior knowledge that is as rough as possible. Given an instance \(x\), the probability of \(x\) belonging to the label \(l_i\) is: \(P(x|l_i) = \frac{|x|_w}{|l_i|_w}\), where \(|x|_w\) denotes the number of keywords appearing in instance \(x\), and \(|l_i|_w\) the total number of keywords about label \(l_i\). To deal with multi-label problems, each instance will have \(k\) probabilities each with respect to a label \(l_i\), it is thus significantly different from the prior model used in Schapire et al. (2002); Wu & Srihari (2004).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>(S)</th>
<th>dim(S)</th>
<th>L(S)</th>
<th>DL(S)</th>
<th>LCard(S)</th>
<th>LDen(S)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(^a)</td>
<td>(^b)</td>
<td>(^c)</td>
<td>(^d)</td>
<td>(^e)</td>
<td>(^f)</td>
</tr>
</tbody>
</table>

\(^a\): number of examples
\(^b\): number of features
\(^c\): number of labels
\(^d\): distinct label sets which counts the number of distinct label sets in \(S\)
\(^e\): label cardinality which measures the average number of labels per example
\(^f\): label density which normalizes LCard(S) by the number of possible labels
**Table 2** Keywords used for the top 10 most frequent classes in the Reuters-21578 dataset

<table>
<thead>
<tr>
<th>class label</th>
<th>keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>earn</td>
<td>share, shr, cts, net, qtr, quarter, profit</td>
</tr>
<tr>
<td>acq</td>
<td>company, acquire, acquisition, merger, stake</td>
</tr>
<tr>
<td>money-fx</td>
<td>currency, bank, dollar, money</td>
</tr>
<tr>
<td>grain</td>
<td>agriculture, wheat, corn, grain</td>
</tr>
<tr>
<td>crude</td>
<td>oil, crude, barrel, petroleum</td>
</tr>
<tr>
<td>trade</td>
<td>export, trade, surplus, import, deficit, tariffs</td>
</tr>
<tr>
<td>interest</td>
<td>rate, bank, money</td>
</tr>
<tr>
<td>wheat</td>
<td>wheat</td>
</tr>
<tr>
<td>ship</td>
<td>industries, subsidiary</td>
</tr>
<tr>
<td>corn</td>
<td>corn</td>
</tr>
</tbody>
</table>

### 3.3 Experimental Setup

As shown in Section 1, there have been many algorithms to deal with multi-label learning problems. In this paper, to evaluate the performance of our proposed method, we compare KnowBoost.MH with the boosting-style algorithm AdaBoost.MH (Schapire & Singer, 1999, 2000), multi-label kernel method Rank-SVM (Elisseeff & Weston, 2002) and multi-label lazy learning algorithm ML-KNN (Zhang & Zhou, 2007), which are all state of the art algorithms applicable to various multi-label learning problems. For AdaBoost.MH, the number of boosting rounds is set to be 500 because its performance will not significantly change after the specified boosting rounds. For Rank-SVM, polynomial kernels with degree 8 are used which yield the best performance as shown in the literature (Elisseeff & Weston, 2002). For ML-KNN, Euclidean metric is used to measure distances between instances and the number of nearest neighbors is set to be 10 as shown in the literature Zhang & Zhou (2007).

### 3.4 Evaluation Criteria

Before presenting comparative results of each algorithm, evaluation criteria in multi-label learning are introduced in this subsection. Performance evaluation of a multi-label learning system is different from that of a single-label learning system. Five popular multi-label evaluation criteria proposed in Schapire & Singer (2000) are as follows: Hamming Loss, One-Error, Coverage, Ranking Loss and Average Precision. In the rest of this paper, the performance of each multi-label learning algorithm is evaluated based on the above five evaluation criteria.

### 4 Experimental Results and Discussions

#### 4.1 Results

For both Reuters-21578 and Newsgroups datasets, we do the same following operations. 25% data are kept as test examples while the rest are used as
<table>
<thead>
<tr>
<th>class label</th>
<th>keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>alt.atheism</td>
<td>god, atheism, christ, jesus, religion, atheist</td>
</tr>
<tr>
<td>comp.graphics</td>
<td>graphics, color, computer, computers, screen</td>
</tr>
<tr>
<td>comp.os.ms-windows.misc</td>
<td>computer, computers, operating, system, microsoft, windows, ms, dos</td>
</tr>
<tr>
<td>comp.sys.ibm.pc.hardware</td>
<td>computer, computers, ibm, pc, hardware, cpu, disk</td>
</tr>
<tr>
<td>comp.sys.mac.hardware</td>
<td>computer, computers, mac, macintosh, hardware, cpu, disk</td>
</tr>
<tr>
<td>comp.windows.x</td>
<td>computer, computers, windows, unix</td>
</tr>
<tr>
<td>misc.forsale</td>
<td>sale, selling, price</td>
</tr>
<tr>
<td>rec.autos</td>
<td>car, drive, fast, ford, honda, gm, engine</td>
</tr>
<tr>
<td>rec.motorcycles</td>
<td>motorcycle, honda, wheel, engine</td>
</tr>
<tr>
<td>rec.sport.baseball</td>
<td>baseball, hit, ball, base, runs</td>
</tr>
<tr>
<td>rec.sport.hockey</td>
<td>hockey, stick, goal, check</td>
</tr>
<tr>
<td>sci.crypt</td>
<td>cryptography, security, secret, key</td>
</tr>
<tr>
<td>sci.electronics</td>
<td>electronics, computer, computer, computers, chip</td>
</tr>
<tr>
<td>sci.med</td>
<td>medicine, doctor, science, sick</td>
</tr>
<tr>
<td>sci.space</td>
<td>space, nasa, rocket, shuttle</td>
</tr>
<tr>
<td>soc.religion.christian</td>
<td>religion, christian, jesus, christ, god, catholic</td>
</tr>
<tr>
<td>talk.politics.guns</td>
<td>guns, gun, kill, shoot, shot</td>
</tr>
<tr>
<td>talk.politics.mideast</td>
<td>mid, east, israel</td>
</tr>
<tr>
<td>talk.politics.misc</td>
<td>politics, clinton, president, congress</td>
</tr>
<tr>
<td>talk.religion.misc</td>
<td>religion, jewish, christian, catholic, god</td>
</tr>
</tbody>
</table>
the pool of training examples. The test sets are then used to evaluate their performance. The pseudo training sets are always generated by applying the prior model in the test sets. Eight true training sets are created by picking up \( m_i \) examples randomly from the pool of training examples, where \( m_i = 32 \times 2^i, i \in [0, 7] \). KnowBoost.MH is trained on the combination of these true labeled datasets and the pseudo datasets, while other four classifiers on these true labeled datasets. Each experiment is conducted ten times and each experimental result is averaged on the ten times results. Figure 1 and 2 report the experimental results of KnowBoost.MH and other multi-label learning algorithms on the Reuters-21578 and Newsgroups datasets, respectively. Therefore, we can make some observations from Figure 1 and 2:

- KnowBoost.MH have a large improvement in terms of all five evaluation criteria before the training set is increased to 256 (i.e. \( i=3 \)) on the Reuters-21578 dataset. It is worth noting that the performance of KnowBoost.MH will not significantly change after the size of the training set is more than 256 (i.e. \( i=3 \)) on the Reuters-21578 dataset. However, KnowBoost.MH always have a large improvement in terms of all five evaluation criteria on the Newsgroups dataset.

- The improvement of KnowBoost.MH on the Newsgroups dataset is larger than that on the Reuters-21578 dataset.

- On the Reuters-21578 dataset, KnowBoost.MH always effectively improve the performance of AdaBoost.MH in terms of all five evaluation criteria when the training set is smaller than 256 (i.e. \( i=3 \)); on the Newsgroups dataset, KnowBoost.MH also do so in terms of One-Error, Coverage, Ranking Loss and Average Precision when the training set is smaller than 512 (i.e. \( i=4 \)).

- In terms of Hamming Loss, KnowBoost.MH is also comparative to AdaBoost.MH in all situations on the Reuters-21578 and Newsgroups dataset.

- When the training set is larger than 256 (i.e. \( i=3 \)) on the Reuters-21578 dataset, the performance of KnowBoost.MH becomes still comparative to AdaBoost.MH algorithm in terms of all evaluation criteria, while when the training set is larger than 512 (i.e. \( i=4 \)) on the Newsgroups dataset, the performance of KnowBoost.MH is a little worse than that of AdaBoost.MH, but becomes still comparative.

- KnowBoost.MH outperforms Rank-SVM in terms of all five evaluation criteria in all situations on the Reuters-21578 and Newsgroups dataset. KnowBoost.MH outperforms ML-KNN in terms of all five evaluation criteria in all situations on the Reuters-21578 dataset. KnowBoost.MH outperforms ML-KNN in terms of One-Error, Coverage, Ranking Loss and Average Precision in all situations on the Newsgroups dataset. Furthermore, KnowBoost.MH is also comparative to ML-KNN in terms of Hamming Loss in all situations on the Newsgroups dataset.
4.2 Discussions

The above results seem interesting, from which we have the following considerations:

- One contribution of KnowBoost.MH is that for fairly small datasets, using prior knowledge may achieve large performance improvements. The reason is that when the training set is small, classifiers can’t be trained sufficiently, while prior knowledge plays a role to compensate the lack of training data. But when training data is enough for training a good classifier, prior knowledge can’t provide extra useful information especially when the prior knowledge is extracted from the training data.

- The improvement of KnowBoost.MH on the Reuters-21578 dataset is smaller. A most possible reason is that prior knowledge on the Reuters-21578 dataset is not the same good as that on the Newsgroups dataset, which provides only few useful information.

- The performance of KnowBoost.MH on the Newsgroups dataset is a little worse than that of AdaBoost.MH but still comparative when the training set is larger than 512 (i.e. $i=4$). That is because that the noisy part of the pseudo training data slightly hurt its performance.

- KnowBoost.MH only obtains comparative results in terms of Hamming Loss on the Newsgroups dataset. It may be because that Hamming Loss is a stringent evaluation criterion. It is difficult to achieve large improvement of performance on this evaluation criterion.

- Both ML-KNN and Rank-SVM are inferior to KnowBoost.MH on the Reuters-21578 and Newsgroups datasets, respectively. ML-KNN and Rank-SVM don’t obtain good classifiers because of lacking enough training data. This is a common problem of most data-driven learning algorithms.

<table>
<thead>
<tr>
<th>confidence parameter</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.170±0.048</td>
<td>0.101±0.015</td>
<td><strong>0.089±0.020</strong></td>
<td>0.108±0.017</td>
<td>0.121±0.037</td>
<td>0.121±0.037</td>
</tr>
</tbody>
</table>

As mentioned in Subsection 2.2, there is a confidence parameter $\theta$ in the KnowBoost.MH algorithm. Ten-fold cross-validation is performed on a random sample of size 200 of the Reuters-21578 dataset for determining the optimal confidence parameter $\theta$. The experimental results of KnowBoost.MH are described in Table 4, where the value of confidence parameter $\theta$ considered by KnowBoost.MH varies from 0.5 to 1.0 with an interval of 0.1. The value following "$\pm$" shows the standard deviation and the optimal result on hamming loss is shown in bold-face. Table 4 shows that the optimal confidence parameter $\theta$ is 0.7, which is actually adopted in the comparison experiments presented in the subsections.
above. As shown in Table 4, KnowBoost.MH is sensitive to different choices of parameter $\theta$ because of different quality of prior knowledge. The robustness of incorporating prior knowledge with different quality should be considered in the next work.

5 Conclusion

In this paper, a novel approach to multi-label learning named KnowBoost.MH is proposed by incorporating prior knowledge into boosting to compensate for the lack of training examples. KnowBoost.MH is trained on the combination of the true labeled training set and the pseudo training set generated by applying the prior knowledge into the test set. For the pseudo training set, we select the most useful instances as additional training data for enhancing the prediction performance. Experimental results on two real-world multi-label datasets, i.e. Reuters-21578 and Newsgroups, show that our proposed KnowBoost.MH achieves superior performance to other state-of-the-art multi-label learning algorithms, especially in the case of the lack of training data.

The robustness of incorporating prior knowledge with different quality is an interesting future work. In addition, support vector machines and neural networks by using prior knowledge can in principle not only reduce the need for larger training dataset, but also enhance the performance of them, respectively. So, it is interesting to extend them to multi-label learning problems.

Acknowledgement

This work was supported by the Natural Science Foundation of China under the Grant No. 60873129 and 61005006, and the Open Projects Program of National Laboratory of Pattern Recognition in China.

References


KnowBoost.MH for Multi-Label Classification

Learning and Data Mining in Pattern Recognition’, Vol. 2734 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, pp. 251–274.


Figure 1  Statistical results on the Reuters-21578 dataset by using four classifiers
KnowBoost.MH, AdaBoost.MH, ML-KNN, Rank-SVM as the number of
training examples increases. (a) hamming loss; (b) one-error; (c) coverage;
(d) ranking loss; (e) average precision
Xiao Wang and Guo-Zheng Li

Figure 2  Statistical results on the Newsgroups dataset by using four classifiers KnowBoost.MH, AdaBoost.MH, ML-KNN, Rank-SVM as the number of training examples increases. (a) hamming loss; (b) one-error; (c) coverage; (d) ranking loss; (e) average precision