Mining Multi-label Concept-Drifting Data Streams Using Dynamic Classifier Ensemble

Conference Paper · November 2009

DOI: 10.1007/978-3-642-05224-8_24 · Source: DBLP

4 authors, including:

Yang ZHANG
Northwest A & F University
77 PUBLICATIONS **557** CITATIONS

SEE PROFILE

Mining Multi-label Concept-Drifting Data Streams Using Dynamic Classifier Ensemble*

Qu Wei, Zhang Yang, Zhu Junping, and Qiu Qiang

College of Information Engineering, Northwest A&F University
Yangling, Shaanxi Province, P.R. China, 712100
{lex,zhangyang,junpinzhu,qiuqiang}@nwsuaf.edu.cn

Abstract. The problem of mining single-label data streams has been extensively studied in recent years. However, not enough attention has been paid to the problem of mining multi-label data streams. In this paper, we propose an improved binary relevance method to take advantage of dependence information among class labels, and propose a dynamic classifier ensemble approach for classifying multi-label concept-drifting data streams. The weighted majority voting strategy is used in our classification algorithm. Our empirical study on both synthetic data set and real-life data set shows that the proposed dynamic classifier ensemble with improved binary relevance approach outperforms dynamic classifier ensemble with binary relevance algorithm, and static classifier ensemble with binary relevance algorithm.

Key words: Multi-label; Data Stream; Concept Drift; Binary Relevance; Dynamic Classifier Ensemble

1 Introduction

Modem organization generate tremendous amount of data by real-time production systems at unprecedented rates, which is known as data streams [1]. Other than the data volume, the underlying processes generating the data changes during the time, sometimes radically [2]. These changes can induce more or less changes in target concept, and is known as concept drift [3].

For the traditional classification tasks, each example is assigned to a single label l from a set of disjoint class labels L [5], while for multi-label classification tasks, each example can be assigned to a set of class labels $Y \subseteq L$. The problem of mining single-label data streams has been extensively studied in recent years. However, not enough attention has been paid to the problem of mining multi-label data streams, and such data

* This work is supported by Young Cadreman Supporting Program of Northwest A&F University (01140301). Corresponding author: Zhang Yang.

streams are common in the real world. For example, a user is interested in the articles of certain topics over time, which may belong to more than one text category. Similarly, in medical diagnosis, a patient may suffer from multiple illnesses at the same time. During the medical treatments at different periods, some of them are cured while others still remain.

In this paper, an dynamic classifier ensemble approach is proposed to tackle multi-label data streams with concept drifting. We partition the incoming data stream into chunks, and then we learn an improved binary relevance classifier from each chunk. For classifying a test example, the weight of each base classifier in the ensemble is set by the performance of the classifier on the neighbors of the test example, which are found in most up-to-date chunk by k-nearest neighbor algorithm. Our experiment on both synthetic data set and real-life data set shows that the proposed approach has better performance than the comparing algorithms.

The rest of the paper is organized as follows. Section 2 reviews the related works. Section 3 presents our improved binary relevance methods. Section 4 outlines our algorithm for mining multi-label concept-drifting data streams. Section 5 gives our experiment result, and section 6 concludes this paper and gives our future work.

2 Related Works

To the best of our knowledge, the only work on multi-label data stream classification is our previous work [4], while a lot of works on classification of multi-label data sets and classification of single-label data streams could be found in the literature.

Multi-label Classification. Multi-label classification methods could be categorized into two groups [5]: problem transformation methods, and algorithm adaptation methods. The first group of methods transforms the original multi-label classification problems into one or more binary classification problems [6]. The second group of methods extends traditional algorithms to cope with multi-label classification problems directly [6], including decision trees [7], boosting [8], probabilistic methods [9], neural networks and support vector machines [10–13], lazy methods [14, 15] and associative methods [16]. Compared with problem transformation methods, the algorithm adaptation methods is always time-consuming, which makes it not suitable for mining multi-label data streams, as data streams are always characterized by large volume of data.

Data Stream Classification. Incremental or online data stream approaches, such as CVFDT [2], are always time and space consuming.

Ensemble learning is among the most popular and effective approaches to handle concept drift, in which a set of concept descriptions built over different time intervals is maintained, predictions of which are combined using a form of voting, or the most relevant description is selected [17]. Many ensemble based classification approaches have been proposed by the research community [1, 18–22]. Two initial papers on classifying data streams by ensemble methods combine the results of base classifiers by static majority voting [18] and static weighted voting [1]. And later, some dynamic methods [19–22] were proposed. In dynamic integration, each base classifier receive a weight proportional to its local accuracy instead of the global classification accuracy. It is concluded in [22] that dynamic methods perform better than static methods.

Multi-label Data Streams. As a multi-label example can belongs to more than one category simultaneously, single-label data streams could be looked as a particular type of multi-label data streams. Therefore, any single-label data streams can be tackled by multi-label data stream classification approaches. As we mentioned before, multi-label data streams are common in the real-life applications. In our previous work [4], we transform each data chunk from the stream into $|L|$ single-label data chunks by binary relevance method, and an ensemble of classifiers are trained from the transformed chunks, and the classifiers in the ensemble are weighted based on their expected classification accuracy on the test data under the time-evolving environment. Although binary relevance method is effective, it does not take the dependency among labels into account. In this paper, we use the classified label vector as a new feature to help classify the other related labels. In our previous work, we adopt static ensemble method, in this paper, we employ dynamic integrated ensemble method to mining concept drifting multi-label data streams.

3 Binary Relevance

Here, we introduce some notations that are used in the rest of the paper. Let's write (x, y) for an example in the training data set, with $x = \{x_1, x_2, \cdots, x_{|F|}\}^T$ being a $|F|$-dimensional feature vector, and $y = \{y_1, y_2, \cdots, y_{|L|}\}$ being a $|L|$-dimensional label vector.

3.1 Binary Relevance

Binary Relevance (BR) learning is a widely used problem transformation method [5]. For each example, if original example contains label l, $(l \in L)$,

then it is labeled as l in the generated single label data set D_l, and as $\neg l$ otherwise [5]. By this way, BR method transforms the original multi-label data set into $|L|$ single-label data sets $D_1, D_2, ..., D_{|L|}$. For each data set D_l, a binary classifier is trained, and the final labels for each testing example can be determined by combining the classification results of all these $|L|$ binary classifiers.

3.2 Improved Binary Relevance

In [13], Godbole *et al.* proposed the SVM-HF method, which could be considered as an extension to BR [6]. Suppose that $i \in L$ and $j \in L$ are two labels, Godbole *et al.* assume that if i is a good indicator of j, then we can gain better performance on j with the information provided by i [13].

Firstly, for each label l ($l \in L$), a binary classifier B_l is trained on $D_l = \{(x, y_l)\}$, and thus we get the the BR classifier $B = \{B_1, B_2, \cdots, B_{|L|}\}$. Then, for each label l, B_l is used to classify example (x, y_l) in data set D_l, and the output v_l is used as an additional feature, which is added into the original data set. In this way, the original data set is transformed into $D' = \{(x, v_1, v_2, ..., v_{|L|}, y)\}$.

Learning algorithm. In this paper, we made two modification to the SVM-HF method proposed in [13]. We will refer the modified method as the improved BR method. When learning for label l. 1) We set the weights of additional features, which are unrelated to l, to 0, as they provide no useful information for learning label l, and furthermore, they may induce noise information. Here, any certain feature selection algorithm, such as mutual information, and information gain, can be used to determine whether a label is related to l. 2) For a related additional feature i, we set its weight to the training accuracy of B_i on D_i. If B_i has poor classification performance, then there may exist a lot of noise in the classification result of B_i on D_i, which means that the additional feature introduced by label i may not be very helpful for learning l, and therefore it should be assigned with low weight. The detailed algorithm is showed in Algorithm 1.

Classification algorithm. In the testing phase, for a given test example t, firstly, t is classified by B, and the label vector outputted by B is added into t. Then, the final label for t is determined by combining the classification results from all the binary classifiers in IB.

Algorithm 1 The learning algorithm for improved BR method.

Input:
D: the original data set;
L: the set of labels;
Output:
B: the BR classifier;
IB: the improved BR classifier;

```
transform D into |L| single-label data sets, {D_1, D_2, ..., D_|L|};
for each l ∈ L do
    train a binary classifier B_l from D_l;
    V_l=B_l.Classify(D_l);
    add V_l as a new feature to D;
    set the weight of feature V_l in D to the training accuracy of B_l;
end for
B = {B_1, B_2, ..., B_|L|};
transform D into |L| single-label data sets, {D'_1, D'_2, ..., D'_|L|};
for l ∈ L do
    set the weights of new added features which are unrelated to l to 0;
    train a binary classifier IB_l from D'_l;
end for
IB = {IB_1, IB_2, ..., IB_|L|};
return B, IB;
```

4 Mining Multi-label Data Streams by Dynamic Classifier Ensemble

The incoming multi-label data stream could be partitioned into sequential chunks with the same chunk size, $S_1, S_2, \cdots, S_n$, with S_n being the most up-to-date chunk. We train a binary relevance classifier B for each S_i ($i \leq n$), then each chunk S_i is transformed into a data chunk S'_i with $|L|$ additional features, which are the outputs of B on S_i. On each transformed data chunk S'_i, we train an improved BR classifier IB.

In data stream scenario, it is impossible and unnecessary to keep and use all the base classifiers, so we only keep K recently trained classifiers, discarding the old classifiers, as classifiers trained on recent data chunks may have better ability to represent the current concept. Algorithm 2 gives an outline of the dynamic classifier ensemble approach.

In static weighted voting ensemble approach, the weight of each classifier in the ensemble are derived from the most up-to-date chunk, S_n. However, as different testing examples are associated with different classification difficulties, if we use global setting in the ensemble to predict a certain example, the weights may not be appropriate for this example. So it is better to weight the classifier dynamically in the testing phase

Algorithm 2 The learning algorithm for classifying multi-label data streams.

Input:

S_n: the most up-to-date chunk;
EB: the ensemble of previously trained BR classifiers;
EIB: the ensemble of previously trained improved BR classifiers;
K: the maximal capacity of the classifier ensemble;

Output:

EB: the updated ensemble of BR classifiers;
EIB: the updated ensemble of improved BR classifiers;

1: Train a BR classifier, B, and an improved BR classifier, IB, on S_n;
2: **if** $|EB| == K$ **then**
3: Remove the oldest classifier in EB;
4: Remove the oldest classifier in EIB;
5: **end if**
6: EB=$EB \bigcup \{B\}$;
7: EIB=$EIB \bigcup \{IB\}$;
8: **return** EB, EIB;

than in the training phase [22]. Our approach is motivated by this idea, given a test example t, some training neighbors of t are found by k-nearest neighbor method. We derive the weights of the classifiers based on the performance of the classifiers on the the training neighbors in S_n. Here, we argue S_n has the most similar concept with t.

The classification algorithm for classifying multi-label data streams is listed in algorithm 3. In step 1 and 10, the function $FindNeighbors()$ is used to find the neighbors for t by k-nearest neighbor method with Euclidean distance. We get the weight for each classifier in step 3-5 and 12-14, the $Evaluate()$ function in step 5 and 14 can be any multi-label evaluation metrics. In sept 17, the final label vector for t is combined with the outputs of EIB using algorithm 4.

5 Experiments

In this section, we report our experiment results on both synthetic data set and real-life data set. The algorithms are implemented in Java with help of WEKA[1] and Mulan[2] software package.

[1] http://www.cs.waikato.ac.nz/ml/weka/
[2] http://mlkd.csd.auth.gr/multilabel.html

Algorithm 3 The classification algorithm for classifying multi-label data streams.

Input:

t: the testing example;
S_n: the most up-to-date chunk;
S'_n: the transformed most up-to-date chunk;
EB: the ensemble of previously trained BR classifiers;
EIB: the ensemble of previously trained improved BR classifiers;
L: the set of labels;
q: the number of neighbors;

Output:

V': the predicted label vector for t;

1: $Neighbor_t$ = FindNeighbors(S_n, t, q);
2: **for** $EB_i \in EB$ **do**
3: Y_i=EB_i.Classify($Neighbor_t$);
4: Z=GetTrueLabels($Neighbor_t$);
5: $Weight_i$=Evaluate(Y_i,Z);
6: BY_i=IB_i.Classify(t);
7: **end for**
8: V=CombineLabels(BY,$Weight$,L);
9: add V as new features to t
10: $Neighbor'_t$ = FindNeighbors(S'_n, t, q);
11: **for** $EIB_i \in EIB$ **do**
12: Y'_i=EIB_i.Classify($Neighbor'_t$);
13: Z'=GetTrueLabels($Neighbor'_t$);
14: $Weight'_i$=Evaluate(Y'_i,Z');
15: IBY_i=EIB_i.Classify(t);
16: **end for**
17: V'=CombineLabels(IBY,$Weight'$,L);
18: **return** V';

5.1 Synthetic Data

We create synthetic multi-label data streams with drifting concepts based on moving hyperplanes [1, 2]. A hyperplane in d-dimensional space is denoted by [1, 2]:

$$\sum_{i=1}^{d} a_i x_i = a_0 \tag{1}$$

Thus, $|L|$ hyperplanes in d-dimensional space are denoted by:

$$\begin{cases} \sum_{i=1}^{d} a_{1_i} x_i = a_{1_0} \\ \sum_{i=1}^{d} a_{2_i} x_i = a_{2_0} \\ \cdots \\ \sum_{i=1}^{d} a_{|L|_i} x_i = a_{|L|_0} \end{cases} \tag{2}$$

Algorithm 4 Class label combination algorithm by weighted voting.

Input:

IBY: the predicted class labels by classifiers in EIB;
$Weight$: the weights of classifiers in EIB;
L: the set of labels;

Output:

V: the combined label vector;

```
for each l ∈ L do
    neg=pos=0;
    for IBY_i ∈ IBY do
        if IBY_{i_l} == NEG then
            neg = neg+Weight_i;
        else
            pos = pos+Weight_i;
        end if
    end for
    if neg > pos then
        V_l = NEG;
    else
        V_l = POS;
    end if
end for
return V;
```

For label n, examples satisfying $\sum_{i=1}^{d} a_{n_i} x_i \geq a_{n_0}$, $(1 \leq n \leq |L|)$, are labeled as positive, and examples satisfying $\sum_{i=1}^{d} a_{n_i} x_i < a_{n_0}$ as negative. Examples are generated randomly, which distributed uniformly in multi-dimensional space $[0, 1]^d$. Weights a_{n_i}, $(1 \leq i \leq d)$, are initialize by random values in range of $[0, 1]$. We set $a_{n_0} = \frac{1}{2} \sum_{i=1}^{d} a_{n_i}$, $(1 \leq n \leq |L|)$, so that each hyperplane cuts the multi-dimensional space into two parts of the same volume. Thus, for each label, roughly half of the examples are positive, and the other half are negative. Noise is introduced by switching the labels of $p\%$ of the training examples randomly.

There is no dependency among labels in the data streams generated in the above way. However, multi-label classification tasks are always characterized by dependency among different labels [6]. In this subsection, we set $|L| = 4$, and introduce dependency into multi-label data streams in the following way:

$$\begin{cases} \sum_{i=1}^{d} a_{1_i} x_i + \sum_{i=1}^{f} a_{1_{(d+i)}} x_i = a_{1_0} \\ \sum_{i=1}^{d} a_{1_i} x_i + \sum_{i=1}^{f} a_{2_{(d+i)}} x_i = a_{2_0} \\ \sum_{i=1}^{d} a_{3_i} x_i + \sum_{i=1}^{f} a_{3_{(d+i)}} x_i = a_{3_0} \\ \sum_{i=1}^{d} \frac{1}{2}(a_{1_i} + a_{3_i}) x_i + \sum_{i=1}^{f} a_{4_{(d+i)}} x_i = a_{4_0} \end{cases} \tag{3}$$

For label 1, we set $a_{1_{(d+1)}}, a_{1_{(d+2)}}, \cdots, a_{1_{(d+f)}}$ to 0, thus it implies $\sum_{i=1}^{d} a_{1_i} x_i + \sum_{i=1}^{f} a_{1_{(d+i)}} x_i = \sum_{i=1}^{d} a_{1_i} x_i$. For label 2, we set $a_{2_{(d+1)}}$, $a_{2_{(d+2)}}$, $\cdots$, $a_{2_{(d+f)}}$ close to 0, so, we get $\sum_{i=1}^{d} a_{1_i} x_i + \sum_{i=1}^{f} a_{1_{(d+i)}} x_i \approx \sum_{i=1}^{d} a_{1_i} x_i + \sum_{i=1}^{f} a_{2_{(d+i)}} x_i$. Thus, if we know the classification result of label 1, we can use this information to help to classify label 2. As the feature space of label 1 is actually d-dimensional, and the feature space of label 2 is $(d+f)$-dimensional, it is harder to learn the concept for label 2 than label 1. The learning task becomes even harder when there is not enough training examples in each data chunk under data stream scenario. However, as label 1 and label 2 have similar concept, we can build the classifier for class label 2 with the help of the classification result on label 1. The term $\sum_{i=1}^{d} \frac{1}{2}(a_{1_i} + a_{3_i}) x_i$ in the formula of label 4 implies that the concept of label 4 depends on 2 concepts, say, label 1, and label 3.

In our experiments, we set $d = 50$, $f = 100$. Parameter f should have a large value, so that to learn from $(d+f)$-dimension feature space is much harder than to learn from d-dimension space. And for the same reason, d should not have a small value. We set $p = 5$ following [1], and concept drifting is simulated by a series of parameters. We write $chunkSize$ for the count of examples in each data chunk from the training stream; k for the total number of dimensions whose weights are involved in concept drifting; $t \in R$ for the magnitude of the change (every N examples) for weights $a_{n_1}, a_{n_2}, \cdots, a_{n_k}$, $(1 \leq n \leq |L|)$; and $s_j \in \{-1, 1\}$, $(1 \leq j \leq k)$, for the direction of change for weights a_{n_j}. Weights change continuously, i.e., a_{n_j} is adjusted by $s_j \cdot t/N$ after each example is generated. Furthermore, there is a possibility of 10% that the change would reverse direction after every N examples are generated, that is to say, s_j is replaced by $-s_j$ with possibility of 10%. Also, each time the weights are updated, we recomputed $a_{n_0} = \frac{1}{2} \sum_{i=1}^{d} a_{n_i}$, so as to keep the class distribution of positive and negative samples.

In our experiments, we set $k = 10$. Parameter k should not have a very small value, since in that case, concept drift is not obvious, and hence the ensemble size does not matter. On the other hand, for a very large value of k, the concept drift is hard to capture. We set $t = 0.1$, $N = 1000$ following [1]. Information gain algorithm is used to select the additional features which are related to the label. And the number of neighbors, q, is set to 10% of the number of examples in the most up-to-date chunk, S_n. For each of the experiment setting, ten trails of experiments are conducted, and the averaged result is reported here. Furthermore, t-Test is used to

illustrate the significance of performance difference between our proposed algorithm and comparing algorithms.

In the synthetic data set, the number of positive labels are nearly equal to that of negative labels, and some examples are labeled as negative in all the labels, so accuracy [13], precision [13], recall [13] and F_1 [23] are not suit for measuring performance of this data set. Hence, we use hamming loss [8] only, and use it as evaluation matric in Algorithm 3.

Here, we write DB_K, DI_K, and SI_K for the classification result for dynamic classifier ensemble with BR algorithm, dynamic classifier ensemble with improved BR algorithm, and static classifier ensemble with improved BR algorithm, respectively. Here, K represents the maximal capacity of the classifier ensemble. In SI_K, the weight of the base classifier is derived by evaluating it on the most up-to-date chunk. The classifier with the lowest weight is discarded when the ensemble is full.

Experiment with *chunkSize*. In this group of experiment, we set $K = 6$, and Naive Bayes algorithm is used as base classifier. Figure 1 shows *chunkSize* could affects hamming loss of the classifiers. Both large and small chunk size can drive up hamming loss. The ensemble can neither capture the concept drifts occurring inside the large data chunk, nor learn a good classifier with not enough training examples in small data chunk [1]. It is shown that DB algorithm gets less hamming loss with large *chunkSize* than DI and SI does, as it is hard to learn the concept of label 2 and label 4 with small data chunk. It is also shown that DI always outperforms DB, which may suggest the strong ability of DI to use the classification result of label 1 and 3 to learn label 2 and label 4.

Experiment with K. In this group of experiment, we set $chunkSize = 1000$, and Naive Bayes algorithm is used as base classifier. Figure 2 shows when maximal capacity of the ensemble increases, due to the increasing diversity inside the ensemble, the hamming loss drops significantly [1]. It is obviously that DB, DI, and SI will get a minimal hamming loss value when K is large enough.

DI algorithm vs. DB algorithm. In this group of experiment, we compare the classification performance of DI algorithm with DB algorithm. Table 1, 2, and 3 gives the classification result measured in hamming loss for DI and DR with Naive Bayes, C4.5, and SVM as base classifier, respectively. Again, as shown in table 1, 2, and 3, *chunkSize* and K affect hamming loss significantly. And from t-Test, it is shown that with different base classifiers, say, Naive Bayes, C4.5, and SVM, DI always outperforms DB significantly, which may suggest the strong ability of DI to utilize the dependency among different labels.

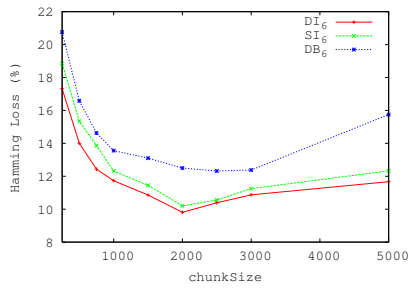

Fig. 1. Experiment with *chunkSize*

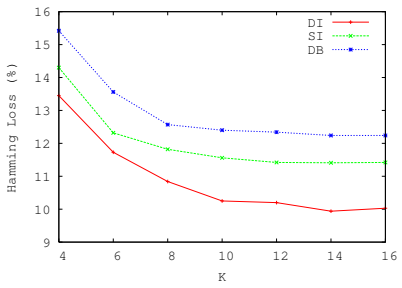

Fig. 2. Experiment with K

Table 1. Hamming Loss(%) of Dynamic Ensemble with Naive Bayes Algorithm

chunkSize	DB_1	DI_1	DB_2	DI_2	DB_4	DI_4	DB_6	DI_6	DB_8	DI_8
250	29.65	**26.45**	29.24	**26.23**	23.55	**20.22**	20.76	**17.32**	18.82	**15.55**
500	24.32	**21.36**	24.22	**21.28**	18.95	**16.12**	16.57	**14.00**	15.07	**12.66**
750	21.74	**19.13**	21.43	**19.06**	16.69	**14.43**	14.63	**12.43**	13.41	**11.34**
1000	19.86	**17.56**	19.72	**17.42**	15.42	**13.45**	13.56	**11.73**	12.57	**10.84**
t-Test	+		+		+		+		+	

Table 2. Hamming Loss(%) of Dynamic Ensemble with C4.5 Algorithm

chunkSize	DB_1	DI_1	DB_2	DI_2	DB_4	DI_4	DB_6	DI_6	DB_8	DI_8
250	35.79	**34.16**	35.12	**33.32**	29.80	**27.34**	27.02	**23.91**	25.23	**21.72**
500	33.20	**31.70**	32.95	**30.98**	27.23	**24.27**	24.64	**21.45**	22.97	**19.76**
750	31.80	**30.24**	31.19	**29.71**	25.83	**23.40**	23.11	**20.39**	21.64	**18.83**
1000	31.16	**29.60**	30.41	**28.94**	25.02	**23.03**	22.59	**20.29**	21.35	**18.71**
t-Test	+		+		+		+		+	

Table 3. Hamming Loss(%) of Dynamic Ensemble with SVM Algorithm

chunkSize	DB_1	DI_1	DB_2	DI_2	DB_4	DI_4	DB_6	DI_6	DB_8	DI_8
250	41.45	**19.13**	38.48	**18.33**	31.41	**16.55**	27.87	**16.43**	25.58	**16.50**
500	23.25	**16.07**	21.39	**15.48**	16.54	**12.45**	14.35	**11.37**	13.29	**10.83**
750	18.48	**13.72**	17.72	**13.24**	13.78	**10.89**	12.25	**10.07**	11.38	**9.59**
1000	16.54	**12.64**	16.08	**12.33**	12.78	**10.49**	11.28	**9.84**	10.54	**9.84**
t-Test	+		+		+		+		+	

Dynamic ensemble vs. static ensemble. In this group of experiment, we compare the classification performance of static classifier ensemble with dynamic classifier ensemble. Here, we use improved BR algorithm. Table 4, 5, and 6 gives hamming loss result with Naive Bayes, C4.5, and SVM as base classifier, respectively. From t-Test, it could be concluded that, dynamic classifier ensemble offers better performance than static classifier ensemble with improved BR algorithm.

Table 4. Hamming Loss(%) of Ensemble Improved BR with Naive Bayes Algorithm

chunkSize	SI_1	DI_1	SI_2	DI_2	SI_4	DI_4	SI_6	DI_6	SI_8	DI_8
250	27.48	**26.45**	27.29	**26.23**	21.53	**20.22**	18.85	**17.32**	17.01	**15.55**
500	22.78	**21.36**	23.25	**21.28**	17.16	**16.12**	15.34	**14.00**	14.38	**12.66**
750	20.65	**19.13**	20.33	**19.06**	16.08	**14.43**	13.86	**12.43**	12.44	**11.34**
1000	18.97	**17.56**	18.85	**17.42**	14.30	**13.45**	12.32	**11.73**	11.82	**10.84**
t-Test	+		+		+		+		+	

Table 5. Hamming Loss(%) of Ensemble Improved BR with C4.5 Algorithm

chunkSize	SI_1	DI_1	SI_2	DI_2	SI_4	DI_4	SI_6	DI_6	SI_8	DI_8
250	35.56	**34.16**	34.66	**33.32**	28.76	**27.70**	24.33	**23.91**	21.99	**21.72**
500	32.78	**31.70**	31.98	**30.98**	25.59	**23.72**	21.25	**21.45**	**19.56**	19.76
750	31.38	**30.24**	30.86	**29.71**	25.03	**23.05**	21.05	**20.39**	18.84	**18.83**
1000	30.35	**29.60**	29.02	**28.94**	24.34	**22.72**	20.42	**20.29**	**18.65**	18.71
t-Test	+		+		+		+			

Table 6. Hamming Loss(%) of Ensemble Improved BR with SVM Algorithm

chunkSize	SI_1	DI_1	SI_2	DI_2	SI_4	DI_4	SI_6	DI_6	SI_8	DI_8
250	21.12	**19.13**	19.02	**18.33**	16.25	**16.55**	16.97	**16.43**	16.95	**16.50**
500	17.54	**16.07**	15.92	**15.48**	12.64	**12.45**	11.35	**11.37**	11.35	**10.83**
750	14.30	**13.72**	13.88	**13.24**	10.81	**10.89**	10.29	**10.07**	9.84	**9.59**
1000	12.65	**12.64**	12.64	**12.33**	10.43	**10.12**	10.01	**9.84**	10.22	**9.84**
t-Test	+		+		+		+		+	

5.2 Real-life Data

In this subsection, we report our experiment result on RCV1-v2[3] text corpus. which is a popular multi-label text data set, and many evaluations have been documented [6] on this corpus. There are 4 main categories in this data set, namely, CCAT, ECAT, GCAT, and MCAT.

As we have shown the ability of the proposed algorithms to cope with gradual concept drift in the previous subsection, here, we preprocess the RCV1-v2 text corpus to simulate sudden concept drift. The scenario involves a user that is interested in different categories over time [24], which is shown in table 7. For example, in period 1, the user is interested in category CCAT and ECAT, and uninterested in GCAT and MCAT; in period 2, he loses his interest in CCAT and begin to show interests in GCAT. By this way, we simulate sudden concept drift.

For each of the experiments, ten trails of experiments are conducted, and the averaged classification performance is reported. We randomly select 49000 examples in each trial. Sudden concept drift occurs after every 6 data chunks, and 49 data chunks are generated with the data from

[3] http://www.jmlr.org/papers/volume5/lewis04a/lewis04a.pdf

Table 7. The User's Interests Over Time

Period	Interested categories	Uninterested categories
1	CCAT,ECAT	GCAT,MCAT
2	ECAT,GCAT	CCAT,MCAT
3	GCAT,MCAT	CCAT,ECAT
4	ECAT,MCAT	GCAT,CCAT
5	CCAT,MCAT	GCAT,ECAT
6	GCAT,CCAT	MCAT,ECAT
7	CCAT,ECAT	GCAT,MCAT
...	...	...

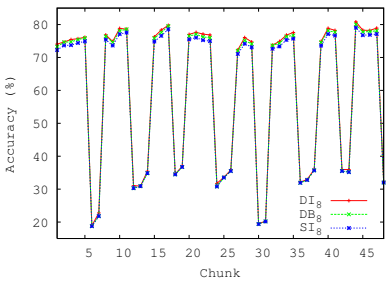

Fig. 3. Accuracy

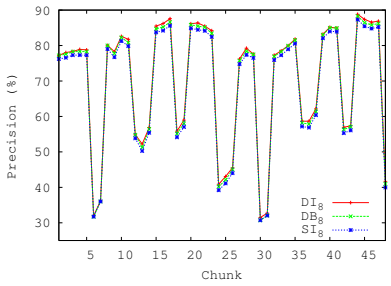

Fig. 4. Precision

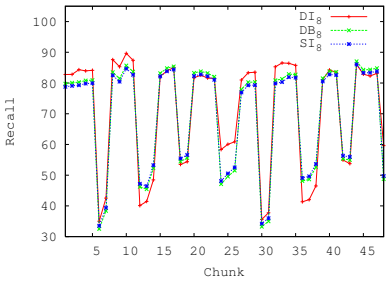

Fig. 5. Recall

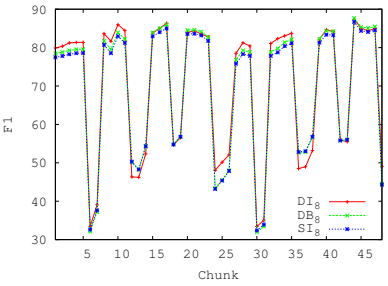

Fig. 6. F_1

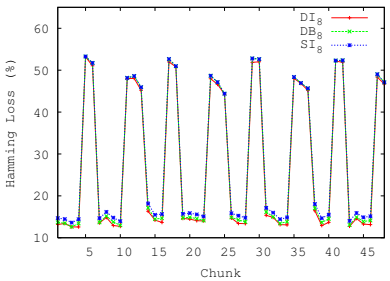

Fig. 7. Hamming Loss

the original text corpus. We set $chunkSize = 1000$, $K = 8$, and Naive Bayes is used as base classifier. We use five different evaluation metrics in algorithm 3 in the five experiments, say, accuracy, precision, recall, F_1, and hamming loss. Figures 3, 4, 5, 6 and 7 show the classification result measured in accuracy, precision, recall, F_1, and hamming loss, respectively. The results show that all the three algorithms, say, DI, DB, and SI, are able to track concept drift, with DI algorithm have slightly better result than DB and SI. It could be observed that the performance of DI is very close to that of DB, this may due to the reason that the dependency among labels in RCV1-v2 is not as strong as that inside our synthetic data set.

6 Conclusion and Future Work

In this paper we propose an improved binary relevance method to take advantage of dependence information among class labels, and propose to classify multi-label data streams with concept drifting by dynamic classifier ensemble with improved BR classifier. The empirical results showed that, comparing with static weighted voting ensemble, and dynamic BR classifier ensemble, dynamic improved BR classifier ensemble method offers better predictive performance. In our future work, we plan to tackle with the multi-label data streams with large label space, since BR classifier will build a binary classifier for each label, it will consume a lot of memory space.

References

1. Wang, H., Fan, W., Yu, P.S., Han, J., H.: Mining concept-drifting data streams using ensemble classifiers. In: Proceeding of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, New York pp. 226–235, 2003.
2. Hulten, G., Spencer, L., Domingos, P.: Mining Time-Changing Data Streams. ACM SIGKDD pp. 97–106, 2001.
3. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts, Machine Learning 23(1):69–101, 1996.
4. Qu, W., Zhang, Y., Zhu, J., Wang, Y.: Mining concept-drifting multi-label data streams using ensemble classifiers. In Fuzzy Systems and Knowledge Discovery, Tianjin, China, 2009. (To appear)
5. Tsoumakas, G., Katakis, I.: Multi-label classification: An overview. Interna-tional Journal of Data Warehousing and Mining, 3(3):1–13, 2007.
6. Tsoumakas, G., Katakis, I., Vlahavas, I.: Mining Multi-label Data, http://mlkd.csd.auth.gr/multilabel.html, Data Mining and Knowledge Discovery Handbook, O. Maimon, L. Rokach (Ed.), Springer, 2nd edition, 2009.

7. Clare, A., King, R.: Knowledge discovery in multi-label phenotype data. In: Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD'01), Freiburg, Germany, pp. 42–53, 2001.
8. Schapire, R.E. Singer, Y.: Boostexter: a boosting-based system for text categorization. Machine Learning, pp. 135–168, 2000.
9. McCallum, A.: Multi-label text classification with a mixture model trained by em. In: Proceedings of the AAAI' 99 Workshop on Text Learning. pp. 681–687, 1999.
10. Crammer, K., Singer, Y.: A family of additive online algorithms for category ranking. Journal of Machine Learning Research, pp. 1025–1058, 2003.
11. Elisseeff, A., Weston, J.: A kernel method for multi-labeled classification. In: Advances in Neural Information Processing Systems. pp. 681–687, 2002.
12. Zhang, M.L., Zhou, Z.H.: Multi-label neural networks with applications to functional genomics and text categorization. IEEE Transactions on Knowledge and Data Engineering, pp. 1338–1351, 2006.
13. Godbole, S., Sarawagi, S.: Discriminative methods for multi-labeled classification. In: Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'04). pp. 22–30, 2004.
14. Zhang, M.L., Zhou, Z.H.: A k-nearest neighbor based algorithm for multi-label classification. In: Proceedings of the 1st IEEE International Conference on Granular Computing. pp. 718–721, 2005.
15. Brinker, K., Hullermeier, E.: Case-based multilabel ranking. In: Proceedings of the 20th International Conference on Artificial Intelligence (IJCAI 07), Hyderabad, India, pp. 702–707, 2007.
16. Thabtah, F., Cowling, P., Peng, Y.: Mmac: A new multi-class, multi-label associative classification approach. In: Proceedings of the 4th IEEE International Conference on Data Mining, ICDM'04. pp. 217–224, 2004.
17. Tsymbal,A.: The problem of concept drift: definitions and related work. In: Technical Report TCD-CS-2004-15, Department of Computer Science, Trinity College Dublin, Ireland, 2004.
18. Street, W., Kim, Y.: A streaming ensemble algorithm (SEA) for large scale classification, in: KDD01, 7th International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, August pp. 377–382, 2001.
19. Zhu, X., Wu, X., Yang, Y.: Dynamic classifier selection for effective mining from noisy data streams. In: Proceedings of the 4th international conference on Data Mining, (ICDM'04), pp. 305–312, 2004.
20. Kolter, J.Z., Maloof, M.A.: Dynamic weighted majority: a new ensemble method for tracking concept drift, in: ICDM03, 3rd International Conference on Data Mining, pp. 123–130, 2003.
21. Zhang, Y., Jin, X.: An automatic construction and organization strategy for ensemble learning on data streams. ACM SIGMOD Record, 35(3):28–33, 2006.
22. Tsymbal, A., Pechenizkiy, M., Cunningham, P., Puuronen. S.: Handling local concept drift with dynamic integration of classifiers: domain of antibiotic resistance in nosocomial infections. In: Proceedings of 19th International Symposium on Computer-Based Medical Systems, (CBMS'06), pp. 679–684, 2006.
23. Gao, S., Wu W., Lee, C.-H., Chua, T.-S.: A maximal figure-of-merit approach to text categorization. SIGIR03, pp.174–181, 2003.
24. Katakis, I., Tsoumakas, G., Vlahavas, I.: Dynamic Feature Space and Incremental Feature Selection for the Classification of Textual Data Streams, ECML/PKDD-2006 International Workshop on Knowledge Discovery from Data Streams, Berlin, Germany, pp. 107–116, 2006.