Classifier Ensemble for Uncertain Data Stream Classification*

Shirui Pan[1], Kuan Wu[2], Yang Zhang[1,**], and Xue Li[3]

[1] College of Information Engineering, Northwest A&F University, China
{panshirui,zhangyang}@nwsuaf.edu.cn
[2] School of Information Science and Technology, Northwest University, China
wukuan1988@163.com
[3] School of Information Technology and Electrical Engineering,
The University of Queensland, Brisbane, Australia
xueli@itee.uq.edu.au

Abstract. Currently available algorithms for data stream classification are all designed to handle precise data, while data with uncertainty or imperfection is quite natural and widely seen in real-life applications. Uncertainty can arise in attribute values as well as in class values. In this paper, we focus on the classification of streaming data that has different degrees of uncertainty within class values. We propose two types of ensemble based algorithms, Static Classifier Ensemble (SCE) and Dynamic Classifier Ensemble (DCE) for mining uncertain data streams. Experiments on both synthetic and real-life data set are made to compare and contrast our proposed algorithms. The experimental results reveal that DCE algorithm outperforms SCE algorithm.

1 Introduction

Mining on streaming data draws more and more attentions in research community in recent years. One of the most challenge issues for data stream mining is classification analysis with concept drift [4,5,6,7,8]. The currently available algorithms for data stream classification are all dedicated to handle precise data, while data with uncertainty is quite common in real-life applications. Uncertainty can appear in attributes in some applications. In a sensor network system, the information such as humidity, temperature and weather usually contains massive uncertainty during the processes of data collecting and transmitting [2]. Uncertainty can also appear in class values. For example, in medical diagnosis, it's hard for the doctor to decide the exact disease of a patient. It's wise to predict all the possibilities of each candidate diseases rather than give a crisp result.

In this paper, we study the data stream classification tasks with uncertain class values. Firstly, we extend the NS-PDT algorithm [1], a decision tree algorithm dealing with categorical attributes and uncertainty in class values, to handle continuous attributes following the scheme of C4.5 [10]. Secondly, we propose two ensemble based algorithms,

* This research is supported by the National Natural Science Foundation of China (60873196) and Chinese Universities Scientific Fund (QN2009092).

** Corresponding author.

M.J. Zaki et al. (Eds.): PAKDD 2010, Part I, LNAI 6118, pp. 488–495, 2010.
© Springer-Verlag Berlin Heidelberg 2010

Static Classifier Ensemble (SCE) and Dynamic Classifier Ensemble (DCE), to classify uncertain data streams with concept drift. Experiments on both synthetic data and real-life data set are made to validate our proposed methods. The experimental results show that DCE outperforms SCE.

This paper is organized as following: Section 2 reviews the related work on data streams and uncertainty. Section 3 describes NS-PDT algorithm for data with uncertainty in class values. Section 4 presents our ensemble based algorithms for classifying uncertain data streams. Section 5 shows our experimental results on both synthetic and real-life data set. And Section 6 concludes this paper and gives our future work.

2 Related Work

To the best of our knowledge, there is no work so far on classifying uncertain data streams, while both studies on mining of data stream and uncertain data have been well investigated in recent years.

At present, the classification algorithms for the data stream scenario are all dedicated to precise data only. A classical approach for data stream classification follows ensemble based scheme [4,5,6,7,8], which usually learns an ensemble of classifiers from the streaming data, and then uses all the classifiers to predict the newly coming instances. The way to integrate classifiers can be distinguished into two categories:

- **Static classifier ensemble.** The weight of each base classifier is decided before the testing phase.
- **Dynamic classifier ensemble.** The weight of each base classifier is decided by the testing instance.

It is concluded in [7] that dynamic methods outperform the static methods [4].

For the classification task on uncertain data, a few decision tree algorithms have been proposed [1,2,3]. Both DTU algorithm [2] and UDT algorithm [3] model the uncertainty in attribute values by probability distribution functions (pdf). This is different from our algorithm, for our goal is to deal with data uncertainty in class values. NS-PDT [1] is a decision tree algorithm to handle data with uncertain class values. However, it can only handle categorical attributes. Hence, in this paper, we firstly extend the initial NS-PDT algorithm [1] to handle continuous attributes and then use it as a base classifier for ensemble classification of data streams.

3 NS-PDT for Uncertain Data

We briefly describe NS-PDT algorithm [1] for uncertainty here. Given an uncertain data set with M classes, the possibility distribution $\pi = \{\pi(L_1), \pi(L_2), \ldots, \pi(L_M)\}$ represents how likely an instance belongs to each class. More specifically, $\pi(L_i)$ accesses the possibility that an instance belongs to class $L_i, i \in [1, M]$.

Compared with the state-of-the-art tree building algorithm C4.5 [10], NS-PDT replaces the Information Gain by the Non-Specificity Gain (NSG), and uses the maximum Non-Specificity Gain Ratio ($NSGr$) to decide which attribute to be selected as the next splitting attribute. NSG is defined as [1]:

$$NSG(T, A_k) = U(\pi^T_{Rep}) - U_{A_k}(\pi^T_{Rep}) \quad (1)$$

Here, $NSG(T, A_k)$ represents the amount of *information precision* obtained after splitting the data set T according to attribute A_k; $U(\cdot)$ accesses the *information precision* of the data set. Please refer to [1] for more details.

As NS-PDT can only deal with categorical attributes, here we simply follow the schemes utilized in C4.5 algorithm [10] to extend NS-PDT to handle continuous attributes. The detailed algorithm is omitted here for lack of space.

4 Mining Uncertain Data Streams

In this paper, we follow the assumption that in data stream scenario, data arrives as batches with variable length [11]:

$$\begin{array}{l} d_{1,1},\ d_{1,2},\ \cdots,\ d_{1,m_1}; \\ d_{2,1},\ d_{2,2},\ \cdots,\ d_{2,m_2}; \\ \cdots; \\ d_{n,1},\ d_{n,2},\ \cdots,\ d_{n,m_n}; \\ \cdots; \end{array} \tag{2}$$

Here, $d_{i,j}$ represents the *j*-th instance in the *i*-th batch. In our paper, we learn a NS-PDT classifier from each batch of uncertain data, and then the base classifiers are combined to form an ensemble. To keep the population capacity of the ensemble, following [6], we delete the oldest classifier when the number of classifiers in the ensemble exceeds a predefined parameter $EnsembleSize$. In the testing phase, our goal is to output a possibility distribution of test instance x over all possible classes.

4.1 Static Classifier Ensemble

We proposed a static classifier ensemble algorithm to classify the uncertain data streams, which is illustrated in algorithm 1.

In algorithm 1, the function $C_i.dis(x)$ in step 3 returns a possibility distribution over different classes of x. In steps 4-6, the possibility for each class is accumulated. In step 8, we normalize the possibility by formula $\pi(L_i) = \frac{\pi(L_i)}{max_{i=1}^{|M|}\{\pi(L_i)\}}$ [1], so that the maximum possibility in π is 1.

Note that in algorithm 1, wei_i represents the weight of classifier C_i, and it is decided in the most up-to-date data batch. Let's write $|NUM|$ for the total number of testing instances; $D(x_i)$ for the accuracy while predicting instance x_i; $\pi^{res}(L_j(x_i))$ for the real possibility distribution of instance x_i on category j; $\pi(L_j(x_i))$ for the predicted resulting possibility distribution of instance x_i on category j. Then the weight wei_i is computed by the PCC_dist metric given by [1]:

$$PCC\_dist = \frac{\sum_{i=1}^{|NUM|} D(x_i)}{|NUM|} \tag{3}$$

$$D(x_i) = 1 - \frac{\sum_{j=1}^{|M|}(\pi^{res}(L_j(x_i)) - \pi(L_j(x_i)))^2}{|M|} \tag{4}$$

Algorithm 1. Classifying Uncertain Data Streams by Static Classifier Ensemble

Input:
E_n : Ensemble of classifiers;
x : Testing instance ;
wei : An array of each classifier's weight;
Output:
$\pi = \{\pi(L_1), \pi(L_2) \ldots, \pi(L_M)\}$: The possibility distribution of x;
1: Initialize $\pi(L_i) = 0, i \in [1, M]$;
2: **for** each $C_i \in E_n$ **do**
3: $pre[] = C_i.dis(x)$;
4: **for** $j = 1$ to M **do**
5: $\pi(L_j) = \pi(L_j) + wei_i \times pre_j$;
6: **end for**
7: **end for**
8: Normalize π;
9: **return** π;

Here, PCC_dist criterion takes into account the average of the distances between the resulting possibility distribution and the real possibility distribution. Note that high value of the PCC_dist criterion implies not only that the algorithm is accurate, but also that the possibility distributions outputted by the algorithm is of high quality and faithful to the original possibility distribution [1].

4.2 Dynamic Classifier Ensemble

In our DCE algorithm, the weight of each base classifier is decided by test instance x. The weighted Euclidean distance between two instances with m attributes is given by :

$$d(x_1, x_2) = \sqrt{\sum_{A_i=1}^{m} w_{A_i} \cdot diff(x_1^{A_i}, x_2^{A_i})^2} \tag{5}$$

For a continuous attribute A_i, we have $diff(x_1^{A_i}, x_2^{A_i}) = \frac{x_1^{A_i} - x_2^{A_i}}{range_{A_i}}$. And for a categorical attribute A_i, we have $diff(x_1^{A_i}, x_2^{A_i}) = \begin{cases} 0 \text{ if } x_1^{A_i} = x_2^{A_i} \\ 1 \text{ if } x_1^{A_i} \neq x_2^{A_i} \end{cases}$. Here, w_{A_i} represents the weight of attribute A_i, and it is decided by the non-specificity gain (NSG) [1] of the attribute A_i. Algorithm 2 gives the details of our DCE algorithm.

In algorithm 2, in step 2 we retrieve K_{nei} neighbors of x from validation data set V according to formula (5). And the most up-to-date data batch from the stream is used as V. In steps 3-9, the weight of each base classifier is determined. The function $C_i.Acc(nei_j)$ in step 6 returns the accuracy of classifier C_i while predicting the instance nei_j, and the accuracy is decided according to formula (4). The function $d(nei_j, x)$ in step 7 returns the distance following formula (5). After predicting an instance, the weight of the corresponding base classifier is accumulated together. Accuracy and distance are the main factors that affect wei_i. In steps 10-14, some base classifiers with lower weight are ignored following [6]. The function $predict(E, x, wei)$ in step 14

Algorithm 2. Classifying Uncertain Data Streams by Dynamic Classifier Ensemble

Input:

E_n : Ensemble of classifiers;
x : Testing instance ;
V : Validation Set;

Output:

$\pi = \{\pi(L_1), \pi(L_2) \ldots, \pi(L_M)\}$: The possibility distribution of x;

```
Initialize π(L_i) = 0, i ∈ [1, M];
Retrieve K_nei neighbors from V to form data set Nei;
for each C_i ∈ E_n do
    wei_i = 0;
    for each nei_j ∈ Nei do
        acc = C_i.Acc(nei_j);
        wei_i = wei_i + acc/d(nei_j, x);
    end for
end for
m_1 = max{wei_i}, i ∈ [1, |E_n|];
m_2 = min{wei_i}, i ∈ [1, |E_n|];
mid = (m_1 + m_2)/2;
E = {e_i | wei_i ≥ mid, e_i ∈ E_n};
π = predict(E, x, wei);
return π;
```

returns the predicted result of x by a weighted voting scheme. The weighted voting scheme is the same as in algorithm 1.

5 Experiments

In this section, we report our experimental results. We deal with the data streams with uncertain class information and concept drift, which is regarded as an important issue in mining data streams [4,5,6,7]. Since no benchmark uncertain data set can be found in the literature, we simulate the uncertain data on synthetic data and real-life data set. Here, moving hyperplane concept [4,7] was used as the synthetic data set with continuous attributes, and RCV1-v2[1] was used as the real-life data set with categorical attributes in our experiments.

The uncertain data set was generated by using the approach described in [1]. We simulated various levels of uncertainty ($X\%$) when generating possibilistic training data. Firstly, we randomly selected $X\%$ of instances in each data batch as uncertain set. Secondly, for each instance in the uncertain set, if the instance belongs to the *i*-th class, we set $\pi(\omega_i) = 1$, and $\pi(\omega_j) = \varphi, \forall j \neq i$. Here, φ is a random value distributed uniformly in range of [0, 1]. Thirdly, for each instance of the remaining $(100 - X)\%$ instances, we assigned a certainly sure possibility distribution (if the instance belongs to the *i*-th class, we set $\pi(\omega_i) = 1$, and $\pi(\omega_j) = 0, \forall j \neq i$). The data set used for test in our experiments was also with a certainly sure possibility distribution, following [1].

[1] http://jmlr.csail.mit.edu/papers/volume5/lewis04a/lyrl2004_rcv1v2_README.htm

PCC_dist is used as the evaluation metric in our experiments for measuring the classification performance of classifiers on uncertain data set. Note that high value of PCC_dist implies not only that the algorithm is accurate, but also that the distribution possibility is faithful to the original possibility distribution [1] .

In our experiments, we set $EnsembleSize$=25, following [7]; and we set K_{nei}=5. Meanwhile, we set $ChunkSize$, the number of instances in each batch, to 500.

Our experiments were made on a PC with Pentium 4 3.0 GHz CPU and 1G memory. All of the algorithms were implemented in Java with help of WEKA[2] software package.

5.1 Moving Hyperplane Concept with Gradual Concept Drift

Moving hyperplanes have been widely used to simulate time-changing concepts [4]. In this subsection, we report our experiments on moving hyperplane concept. A moving hyperplane in d-dimensional space is denoted by: $\sum_{i=1}^{d} a_i x_i = a_0$.

We followed the same procedure in [4] to simulate concept drift. We used K for the total number of dimensions whose weights are changing; S for the magnitude of the change (every N instances) for weights $a_1, \cdots, a_k$. For more detailed descriptions of moving hyperplanes, please refer to [4].

We set d=10, N=1000 in our experiments. For each parameter setting, we generated 100 batches of data. The averaged results on the 100 batches are reported in Table 1.

From Table 1, it could be concluded that, DCE outperforms SCE in all scenarios, with averaged improvement being about 2%. It could be also seen that, with the increasing of X, the performances of both SCE and DCE decline. It is shown that the higher the level of uncertainty in the training set is, the more imprecise the training set will be, and therefore the more difficult for the classifier to learn an accurate model.

We also study the impact of the parameter K_{nei}. Here, we set S=0.1, X=20 and K was selected from [1,10] randomly. We conducted 10 trails and the averaged result is shown in Fig. 1. It can be shown in Fig. 1 that K_{nei} can not be neither too small nor too large. We set K_{nei}=5 in other experiments as it usually gives us good results.

Table 1. Experiments on Moving Hyperplane Concept

X%	S	K=2		K=5		K=8	
		SCE	DCE	SCE	DCE	SCE	DCE
20	0.1	0.770	**0.787**	0.768	**0.785**	0.768	**0.783**
	0.5	0.773	**0.789**	0.764	**0.781**	0.763	**0.778**
	1.0	0.771	**0.790**	0.761	**0.778**	0.762	**0.777**
40	0.1	0.749	**0.767**	0.748	**0.766**	0.748	**0.764**
	0.5	0.749	**0.768**	0.743	**0.760**	0.743	**0.759**
	1.0	0.752	**0.770**	0.738	**0.756**	0.741	**0.758**
60	0.1	0.725	**0.744**	0.723	**0.741**	0.720	**0.738**
	0.5	0.725	**0.744**	0.715	**0.733**	0.717	**0.735**
	1.0	0.726	**0.746**	0.712	**0.732**	0.717	**0.735**

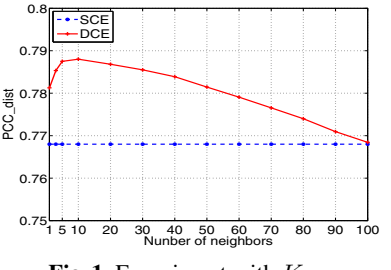

Fig. 1. Experiment with K_{nei}

[2] http://www.cs.waikato.ac.nz/ml/weka/

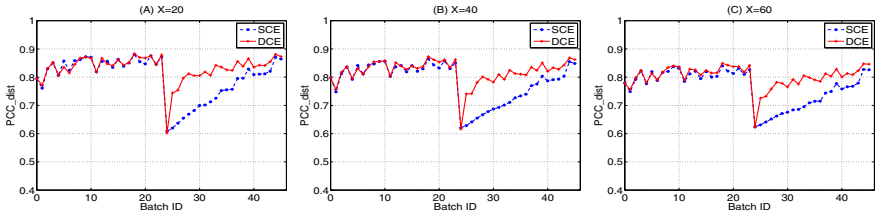

Fig. 2. Experiments on RCV1_v2 Set for Abrupt Concept Drift

5.2 Experiments on RCV1-v2 Text Data Set

In this section, we report our experimental results on a real-life text dataset. RCV1-v2 is a new benchmark collection for text categorization [9]. The news stories in RCV1-v2 were divided into two datasets, training set and testing set. The training dataset was used and four largest categories, CCAT, ECAT, GCAT, and MCAT were considered as the main topics in our experiments to simulate concept drift. After preprocessing, each document was presented by a binary vector, and information gain algorithm was used to select 100 most predictive features.

The data set in our experiments was decomposed into 46 batches. We vary uncertainty $X\%$ as follow: 20, 40, 60. In each scenario of our experiments, we selected one category as *TopicA* and others as *TopicB*. Concept drift occurs between *TopicA* and *TopicB*. Therefore, there are 12 possible combination for 4 categories. We experiment 12 trails and the averaged result is reported.

We simulate abrupt drift in our experiments as follows: In the batches 0-24, all the instances of *TopicA* were labeled as positive, others were labeled as negative. In batches 25-46, all the instances of *TopicB* were labeled as positive, others were labeled as negative. The experimental results for abrupt concept drift are given in Fig. 2.

As expected, from Fig. 2, it could be seem that in batch 25 when abrupt drift occurs, both DCE and SCE have a dramatic decline in PCC_dist. However, DCE recovers much faster than SCE in the later batches. In all levels of uncertainty, DCE outperforms SCE. It reveals that DCE outperforms SCE in handling abrupt concept drift with uncertainty.

5.3 Time Analysis

Both SCE and DCE consume the same time in training, because the ways to construct the ensemble is the same. Here we compare the testing time. We generated data streams with varied $ChunkSize$ on hyperplane concept (K=10, S=1). Consider 100 batches of data with X=20. Fig. 3 shows that large $ChunkSize$ offers better performances. The DCE also performs better than SCE. However, the testing time of the two methods on the whole streams is significantly different. From Fig. 4 we can see that with a large $ChunkSize$, DCE consumes much more testing time than SCE. It is because when predicting an instance, DCE needs to traverse the whole validation set to get some nearest neighbors. So it's a tradeoff between good performance and faster responding action.

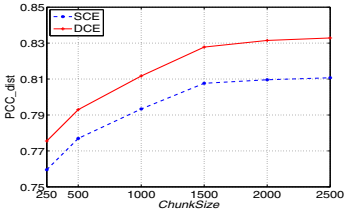

Fig. 3. PCC_dist on varying $ChunkSize$

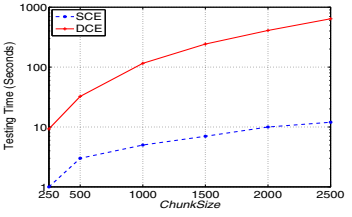

Fig. 4. Testing time on varying $ChunkSize$

6 Conclusion and Future Work

In this paper, we propose two types of ensemble based approaches to classify data streams with uncertainty in class values. We also extend the decision tree (NS-PDT) approach to handle data with continuous attributes. Experiments on both synthetic and real-life datasets are made to validate our proposed methods. Our experimental results show that the dynamic classifier ensemble (DCE) for uncertain data stream outperforms the static classifier ensemble (SCE).

Attribute uncertainty is also natural and prevalent in many real-life applications, we plan to study this kind of uncertain data streams in future work.

References

1. Jenhani, I., Amor, N.B., Elouedi, Z.: Decision trees as possibilistic classifiers. International Journal of Approximate Reasoning 48, 784–807 (2008)
2. Qin, B., Xia, Y., Li, F.: DTU: A Decision Tree for Uncertain Data. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS, vol. 5476, pp. 4–15. Springer, Heidelberg (2009)
3. Tsang, S., Kao, B., Yip, K.Y., Ho, W., Lee, S.D.: Decision Trees for Uncertain Data. In: Proc. of ICDE 2009, pp. 441–444 (2009)
4. Wang, H., Fan, W., Yu, P., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: Proc. of KDD 2003, pp. 226–235 (2003)
5. Zhu, X., Wu, X., Yang, Y.: Dynamic classifier selection for effective mining from noisy data streams. In: ICDM 2004, pp. 305–312 (2004)
6. Zhang, Y., Jin, X.: An automatic construction and organization strategy for ensemble learning on data streams. ACM SIGMOD Record 35(3), 28–33 (2006)
7. Tsymbal, A., Pechenizkiy, M., Cunningham, P., Puuronen, S.: Dynamic integration of classifiers for handling concept drift. Information fusion 9, 56–68 (2008)
8. Kolter, J., Maloof, M.: Dynamic weighted majority: a new ensemble method for tracking concept drift. In: Proc. of ICDM 2003, pp. 123–130 (2003)
9. Lewis, D.D., Yang, Y., Rose, T., Li, F., Zhu, X., Wu, X., Yang, Y.: RCV1: A New Benchmark Collection for Text Categorization Research. Journal of Machine Learning Research 1(5), 361–397 (2004)
10. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufman Publishers, San Francisco (1993)
11. Klinkenberg, R., Joachims, T.: Detecting concept drift with support vector machines. In: Proc. of ICML 2000, pp. 487–494 (2000)