OcVFDT: One-class Very Fast Decision Tree for One-class Classification of Data Streams [*]

Chen Li
College of Information Engineering
Northwest A&F University, China
Yangling, Shaanxi Province, P.R. China, 712100
lichen_0810@nwsuaf.edu.cn

Yang Zhang[†]
College of Information Engineering
Northwest A&F University, China
Yangling, Shaanxi Province, P.R. China, 712100
zhangyang@nwsuaf.edu.cn

Xue Li
School of Information Technology and Electrical Engineering
The University of Queensland, Australia
Brisbane, Queensland 4072 Australia
xueli@itee.uq.edu.au

ABSTRACT

Current research on data stream classification mainly focuses on supervised learning, in which a fully labeled data stream is needed for training. However, fully labeled data streams are expensive to obtain, which make the supervised learning approach difficult to be applied to real-life applications. In this paper, we model applications, such as credit fraud detection and intrusion detection, as a one-class data stream classification problem. The cost of fully labeling the data stream is reduced as users only need to provide some positive samples together with the unlabeled samples to the learner. Based on VFDT and POSC4.5, we propose our OcVFDT (One-class Very Fast Decision Tree) algorithm. Experimental study on both synthetic and real-life datasets shows that the OcVFDT has excellent classification performance. Even 80% of the samples in data stream are unlabeled, the classification performance of OcVFDT is still very close to that of VFDT, which is trained on fully labeled stream.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*data mining*; I.2.6 [**Artificial Intelligence**]: Learning—*concept learning*; I.5.2 [**Pattern Recognition**]: Design Methodology—*classifier design and evaluation*

[†]Corresponding author: Yang Zhang.

[*]This work is supported by Young Cadreman Supporting Program of Northwest A&F University (01140301) and Australian ARC Discovery Project DP0558879.

General Terms

Decision trees, Hoeffding bound, one-class data streams, incremental learning

1. INTRODUCTION

In real world applications, such as credit fraud detection, network intrusion detection, and so on, huge volume of data arrives continuously with high speed. These applications could be modeled as a data stream classification problem. Currently, the research community of data stream classification analysis mainly focus on supervised learning [4][8][11][18][19], which requires fully labeled data streams for training. However, under the data stream scenarios, it is too expensive to label the full stream manually, and therefore the supervised data stream learning algorithms are not applicable to real-life applications.

In the case of credit fraud detection, the user behaviors that cause bad economic effect could be looked as positive samples. For those behaviors which have not caused any bad effect yet, only after thorough investigation could we decide whether they are fraud or not. As it is expensive, and sometimes impossible to investigate their true class labels, it is better to use them as unlabeled samples. The same scenario could be observed in network intrusion detection. In this paper, we model these applications as a problem of one-class classification of data streams.

In one class classification problems, classifiers are trained to distinguish a class of objects (called the target class) from all other objects [23]. In this paper, we identify the following characters of one-class data stream classification:

1. **No negative training samples**: only positive samples and unlabeled samples could be observed from the training data stream.
2. **High speed input data**: the classifier should have the ability to process huge volumes of data which arrive at high speed.
3. **Limited memory space**: only limited memory space is available to the classifier, which means that the classifier has to scan the input samples for only once.

In this paper, based on VFDT [4] and POSC4.5 [2], we present our OcVFDT (One-class Very Fast Decision Tree) algorithm for one-class classification of data streams. The experimental results on both synthetic and real-life dataset show that even with a high percentage of samples in the stream kept unlabeled, the classification performance of OcVFDT is very close to VFDT, which is trained on fully labeled data stream. It still has excellent classification performance even when 80% of the samples in the stream is unlabeled.

This paper is organized as follows. Section 2 reviews the related work. The proposed OcVFDT algorithm is presented in section 3. The detailed experiment setup and results are shown in section 4. Section 5 concludes this paper and gives our future work.

2. RELATED WORK

One-class classification. Current research on one-class classification can be divided into two categories: (1) Methods that focus on the construction of basic classifiers: Schölkopf *et al.* proposed to apply SVM to one-class classification problem [17]. Denis *et al.* proposed one-class decision tree in [2]. Calvo *et al.* gave two methods to enhance the positive naïve bayes (PNB) classifier [1]. Elkan *et al.* assumed that the labeled examples are selected randomly from the positive examples, and gave two approaches for one-class learning [5]. (2) Methods that convert the one-class classification problem into a traditional supervised learning problem or semi-supervised learning problem: In [7][15][21], for text classification, negative text documents are extracted from the unlabeled documents, and supervised leaner is trained with help of these negative documents. In [13], Lee *et al.* transformed all the unlabeled samples into negative samples, and used a linear function to learn from the noisy examples. In [20], Yu proposed MC (Mapping Convergence) method to incrementally label negative data from unlabeled data using the *margin maximization* property of SVM.

Data Stream Classification. There are two main approaches for classifying data streams: single classifier based approach and ensemble based approach. For single classifier based approach, the most well-known classifiers are VFDT [4][10] and CVFDT [11]. The CVFDT improves the VFDT so that it can deal with concept drift. After that, many decision tree based algorithms were proposed for data stream classification [9][12]. For ensemble based approaches, the initial papers use static majority voting [18][19], while the current trend is to use dynamic classifier ensembles [22][24]. Besides, Gama *et al.* presented UFFT algorithm which generated a forest of trees from data streams [8].

One-class classification of Data Streams. In [23], Zhang *et al.* proposed the problem of one-class classification of data streams. To the best of our knowledge, this is the only work devoted to one-class classification of data streams. However, the algorithm proposed in [23] could only tackle text streams, while the algorithm proposed in this paper could cope with general data streams, including text streams. The algorithm in [23] follows the ensemble based approach to cope with concept drift in the stream, while the algorithm in this paper follows the single classifier approach. However, in this paper, concept drift is not considered, and is left for our future research.

3. BUILDING OcVFDT

In this paper, we only consider data streams with discrete attribute values. For data streams with continuous attribute values, the continuous attributes could be discretized to discrete attributes [6], so as to transform the original data stream into a data stream with discrete attributes only.

Given a stream data $S = (s_1, s_2, \ldots, s_i, \ldots)$, where s_i is a sample in S, $s_i =< X_i, y_i, l_i >$. Here, X_i is an attribute vector with n discrete attributes; $y_i \in \{-1, +1\}$, represents the class label of X_i; $l_i \in \{0, 1\}$, represents whether y_i is available or not to the learner. Under the one-class classification scenario, only positive samples are labeled. So if $l_i = 1$, we are sure that $y_i = +1$; and if $l_i = 0$, the true class label of X_i is unknown.

For simplicity, in this paper, concept drift is not considered. And we assume that the positive training samples and unlabeled samples are distributed uniformly in the data stream. If the positive training samples and/or unlabeled samples distribute unevenly in the stream, then concept drifting is presented in this stream. We will study this kind of stream in our future work.

3.1 One-class Information Gain

The information gain algorithm is widely used in decision tree algorithms to decide which attribute being selected as the next splitting attribute [4][16]. Denis *et al.* proposed POSC4.5 algorithm and demonstrated how to measure one-class information gain for static datasets [2]. Here, based on POSC4.5, for the input sample s_i in data stream S, and attribute set $A = \{A_1, A_2, \ldots, A_j, \ldots, A_{|A|}\}$, formula (1) is used to measure one-class information gain for the current tree node, denoted by *node*:

$$OcIG(A_j) = Entropy(s_i) - \sum_{a \in A_j} \frac{|s_{i,a}|}{i} Entropy(s_{i,a}) \quad (1)$$

Here, $Entropy(s_i)$ and $Entropy(s_{i,a})$ represent the entropy of the set $\{s_1, s_2, \ldots, s_i\}$, and set $s_{i,a} = \{s | s \in \{s_1, s_2, \ldots, s_i\}, s_i(A_j) = a\}$, respectively, which is computed by the following formulas:

$$p_{1,set} = \min\{\frac{|POS_{node,set}|}{POS_{i,set}} \times PosLevel \times \frac{|UNL_{i,set}|}{|UNL_{node,set}|}, 1\}$$

$$p_{0,set} = 1 - p_{1,set} \quad (2)$$

$$Entropy(set) = -p_{1,set} \log_2 p_{1,set} - p_{0,set} \log_2 p_{0,set} \quad (3)$$

Here, we write $|POS_{node,set}|$ and $|POS_{i,set}|$ for counting the positive samples in *set* observed at the *node*, and the first i samples in S respectively; $|UNL_{node,set}|$ and $|UNL_{i,set}|$ for counting the unlabeled samples in *set* observed at the *node*, and the first i samples in S respectively. Please note that in order to collect these statistical data, we only need to update some counters whenever a new sample arrives. For the sake of efficiency and limited memory available, we do not save all the samples observed from the stream for multiple scans.

The $PosLevel$ in formula (2) is to estimate the probability of the observed positive samples from the data stream S. For a given classification task, $PosLevel$ is always unknown to the learner. The next subsection will give more discussions on this parameter.

3.2 Building OcVFDT

In [4], Domingos *et al.* proposed to choose the best split attribute when constructing decision tree for data streams based on Hoeffding bound. Here, based on Hoeffding bound, we use the one-class information gain as the split evaluation function to choose the best splitting attribute. As the true value of $PosLevel$ in formula (2) is unknown, we enumerate nine possible values of $PosLevel$, from 0.1 to 0.9. Then, we get a forest, T, with nine different OcVFDTs.

The best tree in T is then chosen by estimating the classification performance of the trees with a chunk of validating samples. The validating chunk is filled with validating samples which are selected randomly with probability $p_{validate}$ from S. Note that due to the limited memory available, we cannot save all the validating samples that have been selected. Once the validating chunk is full, the samples inside it will be used to evaluate the trees in T, and then the validating chunk is cleared. Finally, these evaluation data will be used to choose the best tree as the final output classifier. Our algorithm for building one-class very fast decision tree is illustrated in Algorithm 1.

Algorithm 1 Building one-class very fast decision tree

```
Input:
    a stream of samples, $S$;
    the size of validating chunk, $n_{validate}$;
    the probability for a sample to be selected as validating sample, $p_{validate}$;
    the feature space of stream $S$, $A$;
    one minus the desired probability of choosing the correct attribute at any given node, $\delta$;
    user-supplied tie threshold, $\tau$;
    the number of samples between checks for growth, $n_{min}$.
Output:
    a one-class very fast decision tree.
1: $ValidateChunk = \phi, T = \phi$;
2: for each $i \in [1, 9]$ do
3:     initialize a tree $T_i$ with only a leaf (the root);
4:     $T = T \bigcup T_i$;
5: end for
6: for each sample $s \in S$ do
7:     for each $i \in [1, 9]$ do
8:         $PosLevel = \frac{i}{10}$;
9:         $Grow(T_i, PosLevel, s, A, \delta, \tau, n_{min})$;
10:    end for
11:    if $Random() \leq p_{validate}$ then
12:        $ValidateChunk = ValidateChunk \bigcup \{s\}$;
13:        if $|ValidateChunk| == n_{validate}$ then
14:            $Estimate(T, ValidateChunk)$;
15:            $ValidateChunk = \phi$;
16:        end if
17:    end if
18: end for
19: if $ValidateChunk! = \phi$ then
20:     $Estimate(T, ValidateChunk)$;
21: end if
22: return $GetBestTree(T)$;
```

In this algorithm, the system is initialized in steps 1 to step 5; a forest of 9 trees is trained on S in steps 7 to step 10; the validating chunk is maintained and the trees are evaluated on it in step 11 to steps 21. The function $Grow(T_i, PosLevel, s, A, \delta, \tau, n_{min})$ is used to grow a single one-class decision tree, please refer to section 3.3 for details. The function $Random()$ returns a random value in $[0, 1]$. The function $Estimate(T, ValidateChunk)$ is used to evaluate the classification performance of trees in T on the validating chunk. And the function $GetBestTree(T)$ returns the best tree selected from forest T. These two functions are demonstrated in section 3.4.

3.3 Growth of Single OcVFDT

Based on the growth algorithm of VFDT [4], the process of growing single OcVFDT is listed in Algorithm 2.

Algorithm 2 $Grow(T_i, PosLevel, s, A, \delta, \tau, n_{min})$

```
// Refer to Algorithm 1 for the details of input parameters;
1: $node = T_i.sort(s)$;
2: $n_{node} = getNumOfSamplesAtNode(node)$;
3: Set the class label of $node$ to the majority class at $node$ according to $p_1$ and $p_0$ following formula (2);
4: if $numOfClassesAtNode(node) > 1$ and $n_{node} \% n_{min} == 0$ then
5:     for each $A_i \in node.A$ do
6:         Compute $G_{node}(A_i) = OcIG(A_i)$ following formula (1);
7:     end for
8:     Choose attribute $A_a$ and $A_b$ with the highest and second-highest $G_{node}$;
9:     $\Delta G_{node} = G_{node}(A_a) - G_{node}(A_b)$;
10:    Compute $\epsilon$ following the Hoeffding bound [4];
11:    if $\Delta G_{node} > \epsilon$ or $\Delta G_{node} \leq \epsilon < \tau$ then
12:        for each $a_i \in A_a$ do
13:            $node_i = addChildLeaf()$;
14:            Set the class label of $node_i$ to the majority class at $node_i$ according to $p_1$ and $p_0$ following formula (2);
15:        end for
16:    end if
17: end if
```

Algorithm 2 could be summarized into two main processes. Firstly, from steps 1 to 3, the sample s is traversed along the tree T_i to reach leaf $node$, and the class label of $node$ is set to the majority class at $node$. Secondly, from steps 4 to 17, new leaves are grown at $node$. For each available attributes at $node$, the one-class information gain is derived using formula (1). If the condition at step 11 is satisfied, then $node$ should be split by attribute A_a, and new leaves should be generated. Here, note that when computing the majority class at new leaves, the count of positive and unlabeled samples comes from the parent node.

Let's write v for the maximum number of values per attribute, c for the number of classes, l for the maximum depth of the tree $l < |A|$, and $|S|$ for the number of samples in data stream S, the time complexity of Algorithm 1 is $O(|S||A|vcl) + O(|S||A|vclp_{validate}) = O(|S|)$.

3.4 Tree Selection

A forest T with nine trees is trained in Algorithm 1. Based on the POSC4.5 [2] algorithm, we design our algorithm for choosing the best tree as the final output classifier.

We collect the following statistical data with the help of the validating chunk of each tree $T_k \in T$.

1. $|POS|$: the count of positive samples in the validating chunk;

2. $|UNL|$: the count of unlabeled samples in the validating chunk;

3. $|PToU_k|$: the count of positive samples that are classified as unlabeled samples by T_k;

4. $|UToP_k|$: the count of unlabeled samples that are classified as positive samples by T_k.

Once the validating chunk is full, the function $Estimate(T, ValidateChunk)$ will be invoked to collect the statistical data. And the counting data among different runs should be accumulated together.

The following formula is used to evaluate the performance of each $T_k \in T$:

$$e(T_k) = \frac{|PToU_k|}{|POS|} + \frac{|UToP_k|}{|UNL|} \tag{4}$$

And the best tree is chosen by:

$$T_j = \operatorname*{argmin}_k (e(T_k)) \qquad 1 \leq k \leq 9 \tag{5}$$

4. EMPIRICAL STUDY

In order to measure the classification performance of our OcVFDT algorithm, we perform experiments on both synthetic dataset and real-life dataset. We compare the classification of OcVFDT with VFDT, as both OcVFDT and VFDT are the single-classifier based approaches. Furthermore, as VFDT is a supervised learner, this comparison can also help reveal the strong ability of OcVFDT to learn from unlabeled samples.

Our algorithms are implemented in Java language based on the WEKA[1] software packages, and the experiments are conducted on a PC with Core 2 CPU, 1G memory and Windows XP OS.

We measure the classification performance of the proposed classifier by accuracy and F1, which are commonly used in the research community for measuring the performance of one-class classifiers [2][5].

4.1 Synthetic Data

For each of the experiments settings here, twenty trails of experiments on synthetic data are conducted, and the averaged classification performance is reported.

In the rest of this paper, we write $PosLevel$ for the percentage of positive samples in the stream.

Similar to the experiment setting in [4], we set $\tau = 0.05$, and $\delta = 0.0000001$ for OcVFDT in all experiments.

4.1.1 Generating Synthetic Data Stream

We modify the software in VFML[2] package, which is used to generate synthetic data stream for VFDT [4], by adding a new parameter, $UnLevel$. $UnLevel$ represents the percentage of unlabeled samples in the stream. For each sample in the stream S, there are 100 discriminative attributes with binary attribute value, and 1 category attribute. The testing data stream is set to have the same data distribution and the same size as the training data stream.

[1]http://www.cs.waikato.ac.nz/ml/weka/
[2]http://www.cs.washington.edu/dm/vfml/

4.1.2 Analysis of Parameters in OcVFDT

Parameter $n_{validate}$ and $p_{validate}$. Two groups of experiments are conducted here, with $n_{validate}$ ranging from 200 to 500 in one group, and $p_{validate}$ ranging from 0.3 to 0.6 in the other. We set $|S| = 10^5$ and $UnLevel = 80\%$. The experiment result shows that, for both F1 and accuracy, the difference of classification performance among different experiment settings is non-significant on t-Test [3]. The detailed experiment result is omitted here for lacking of space. We set $n_{validate} = 200$, and $p_{validate}$=0.3 in all experiments for the sake of limited memory space.

Parameter n_{min}. We experimented with $|S| = 10^4$, $|S| = 10^5$ and $|S| = 10^6$, respectively, with $UnLevel = 80\%$, and $PosLevel = 50\%$. Please refer to Fig.1 for the experiment results. In Fig.1, the horizontal axis represents n_{min}, the vertical axis represents classification performance. Fig.1(A), Fig.1(B), and Fig.1(C) gives the experiment result for $|S| = 10^4$, $|S| = 10^5$, and $|S| = 10^6$, respectively.

It can be observed from Fig.1 that with increasing of n_{min}, the classification performance is decreasing. Hence, for the rest of this paper, we set $n_{min} = 200$.

4.1.3 OcVFDT vs. VFDT

Comparison of Classification Performance. In this group of experiments, we compare the classification performance of VFDT and OcVFDT. For VFDT, we set $\tau = 0.05$, $\delta = 0.0000001$, and $n_{min} = 200$.

We experimented with $|S| = 10^4$, 10^5, 10^6 and 10^7; *PosLevel* = 40%, 50%, and 60%; $UnLevel$ = 55%, 60%, 65%, 70%, 75%, and 80%. The experiment results are shown in Fig.2 and Fig.3. In Fig.2 and Fig.3, the horizontal axis represents $|S|$, the vertical axis represents accuracy, and F1, respectively. The different lines in these figures represent the classification result of VFDT, and OcVFDT with various $UnLevel$ parameter.

It could be observed from Fig.2 and Fig.3 that for a certain $|S|$ value, the classification performance of OcVFDT with different $UnLevel$ is very close to each other. It could also be observed that OcVFDT performs a little worse than VFDT, and for some $|S|$ value, it even performs better than VFDT. When $|S| = 10^7, UnLevel = 80\%$, the averaged accuracy of VFDT is 0.961, while the averaged accuracy of OcVFDT is 0.953. Compared with VFDT, the decrease in accuracy of OcVFDT by 0.008 is compensated by freeing the human power to label 8×10^6 samples. This is a great achievement of saving human power, and makes OcVFDT more applicable to real-life applications.

Experiment on Unlabeled Samples. In this group of experiments, we test the ability of OcVFDT to cope with unlabeled data. We set $PosLevel = 50\%$ for generating data stream S. When experimenting with VFDT, we sample $|POS|$ positive samples and $|POS|$ negative samples from S to form the training data stream. When experimenting with OcVFDT, we use $|POS|$ positive samples from S as labeled sample, and $|POS|+|S|(i-1)/10$ samples from S as unlabeled samples to form the training data stream. Here, when we are generating training data streams S for both VFDT and OcVFDT, we make sure that $PosLevel = 50\%$ in the generated stream, and keep the order among samples in S to the generated data stream. By using this method, the assumption of even distribution of original data is broken which leads the performance of OcVFDT not so well.

Tab.1 shows the experiment results. In Tab.1, column

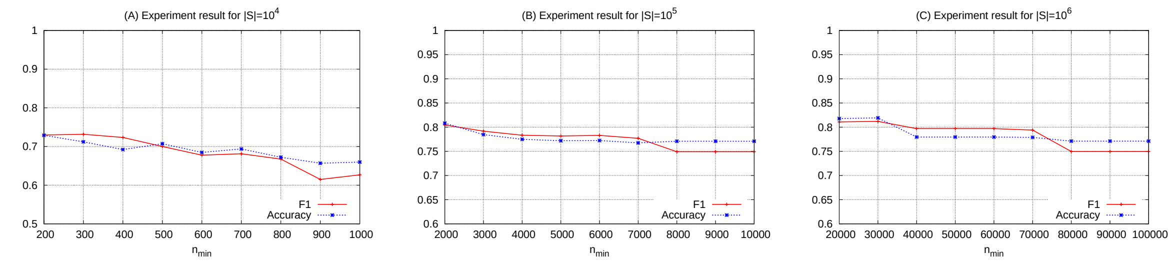

Figure 1: Experiment with n_{min}.

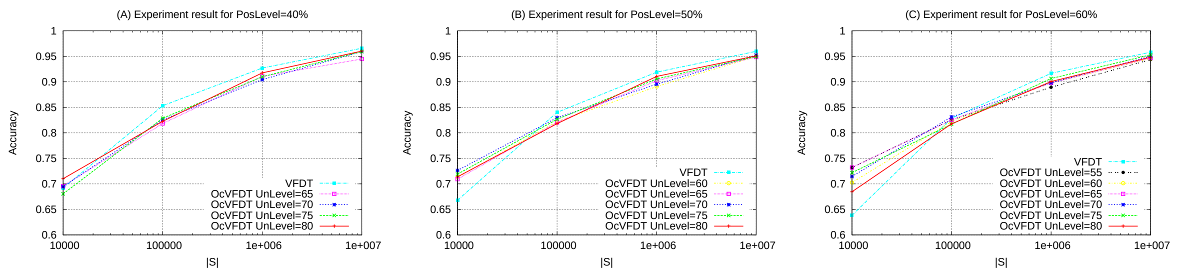

Figure 2: Comparison of VFDT and OcVFDT Measured by Accuracy.

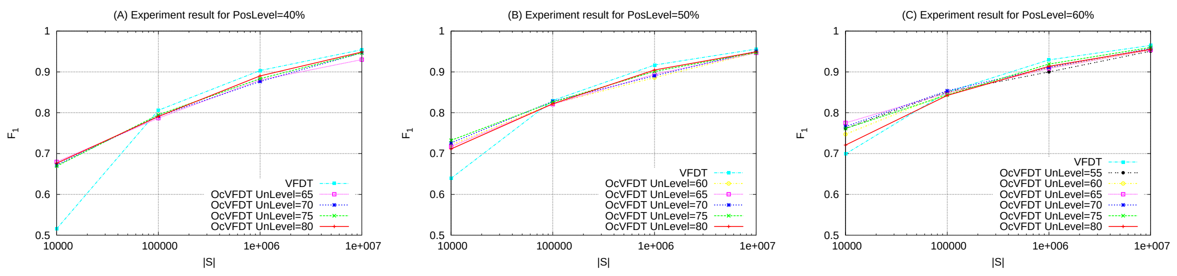

Figure 3: Comparison of VFDT and OcVFDT Measured by F1.

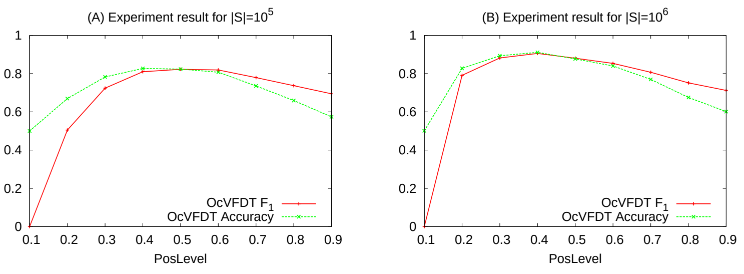

Figure 4: Experiment with Tree Selections.

Table 1: Results of Different Sizes of Unlabeled Samples.

$\|S\|$	$\|POS\|$	VFDT		OcVFDT													
				$i = 1$		$i = 2$		$i = 3$		$i = 4$		$i = 5$		$i = 6$		$i = 7$	
		F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc
10^4	2000	0.606	0.614	0.668	0.501	0.668	0.501	0.668	0.501	0.667	0.607	0.689	0.625	0.721	0.670	0.729	0.715
	2500	0.603	0.618	0.667	0.501	0.667	0.501	0.665	0.557	0.667	0.586	0.708	0.664	0.709	0.706	-	-
	3000	0.572	0.621	0.666	0.500	0.660	0.552	0.667	0.579	0.713	0.651	0.733	0.710	-	-	-	-
	3500	0.522	0.631	0.660	0.512	0.673	0.573	0.690	0.630	0.711	0.696	-	-	-	-	-	-
	4000	0.507	0.636	0.694	0.605	0.719	0.669	0.742	0.734	-	-	-	-	-	-	-	-
10^5	20000	0.786	0.788	0.706	0.597	0.733	0.660	0.749	0.683	0.779	0.746	0.789	0.759	0.801	0.779	0.824	0.818
	25000	0.807	0.798	0.732	0.661	0.741	0.665	0.773	0.730	0.791	0.765	0.804	0.791	0.822	0.825	-	-
	30000	0.811	0.812	0.697	0.572	0.787	0.757	0.799	0.780	0.801	0.781	0.825	0.824	-	-	-	-
	35000	0.817	0.824	0.774	0.731	0.792	0.769	0.793	0.765	0.824	0.822	-	-	-	-	-	-
	40000	0.824	0.832	0.785	0.758	0.788	0.766	0.824	0.824	-	-	-	-	-	-	-	-
10^6	200000	0.891	0.894	0.845	0.831	0.827	0.795	0.865	0.852	0.868	0.858	0.884	0.877	0.902	0.902	0.906	0.917
	250000	0.893	0.896	0.800	0.752	0.867	0.856	0.870	0.861	0.873	0.862	0.896	0.895	0.901	0.905	-	-
	300000	0.897	0.899	0.862	0.849	0.879	0.875	0.859	0.848	0.895	0.896	0.897	0.902	-	-	-	-
	350000	0.901	0.903	0.876	0.871	0.850	0.834	0.894	0.895	0.897	0.904	-	-	-	-	-	-
	400000	0.910	0.913	0.835	0.812	0.885	0.883	0.885	0.890	-	-	-	-	-	-	-	-

Table 2: Running Time and Size of the Tree.

$\|S\|$	VFDT			OcVFDT		
	Time(ms)	#Leaves	#Node	Time(ms)	#Leaves	#Node
20000	91.00	8.6	7.6	680.84	11.2	10.2
200000	691.92	64.4	63.4	8004.48	65.8	64.8
2000000	7335.68	313.0	312.0	106935.90	441.8	440.8
20000000	78450.00	1529.0	1528.0	1459911.00	3775.0	3774.0

1 lists the size of data stream S; column 2 gives the size of (labeled) positive sample set, $|POS|$; columns 3 and 4 give the experiment results of VFDT, measured in F1 and accuracy respectively; columns 5 to 18 give the experiment results of OcVFDT with different i values. It is shown in Tab.1 that from $i = 1$ to 7, the classification performance of OcVFDT is improving, because the total number of training samples is increasing and the distribution of the training set is becoming more and more uniform.

4.1.4 *Experimentation of Tree Selections*

In this group of experiments, we examine the ability of our tree selection algorithm for choosing the best trees. When generating data streams, we set $PosLevel = 50\%$, and $UnLevel = 80\%$. Experiment results of $|S| = 10^5$ and $|S| = 10^6$ are reported in Fig.4(A) and Fig.4(B) respectively. In Fig.4, the horizonal axis represents nine OcVFDTs trained with nine different $PosLevel$ values, and the vertical axis represents the classification performance. In Fig.4, "T_i *F1*", "T_i *Accuracy*", "*OcVFDT F1*", and "*OcVFDT Accuracy*" represents the F1 index of tree T_i, the accuracy index of T_i, the F1 index of the selected tree, and the accuracy index of the selected tree respectively.

It is shown in Fig.4 that our tree selection algorithm is very effective in choosing the best tree from the forest T and in the 20 trails experiments both in $|S| = 10^5$ and $|S| = 10^6$. For $|S| = 10^5$, there are 7 trails which select the top tree, 10 trails select the second tree and 3 trails choose the third tree. For $|S| = 10^6$, all the trails select the top tree. For lack of space in this paper, we can't report the results of other $UnLevel$ values.

4.1.5 *Running Time and Size of the Tree*

Here, we report the running time, the count of inner nodes, and the count of leaves of VFDT and OcVFDT with different $|S|$ parameters. When generating data streams, we set $PosLevel = 50\%$, $UnLevel = 80\%$. The running time reported here is the time for the leaner to learn from the samples. The time for input/output is not considered.

The experiment results are listed in Tab.2. In Tab.2, column 1 lists the size of the data stream, $|S|$; columns 2, 3, and 4 list the running time, the count of leaf nodes, the count of inner nodes for VFDT respectively; and columns 5,6, and 7 give these information for OcVFDT.

4.2 Real-life Data

We perform experiments on one real-life dataset, namely Reuters Corpus Volume 1 (RCV1) [14].

4.2.1 *Datasets and Preprocessing*

RCV1. Reuters Corpus Volume 1 (RCV1) consists of English language stories produced by Reuters journalists between August 20, 1996 and August 19, 1997. To make it easier to use, Lewis *et al.* provided RCV1-V2 [3] after preprocessing on RCV1, which becomes a benchmark dataset [14] for text mining researchers.

There are four main categories in RCV1-V2, i.e., ECAT, GCAT, CCAT, and MCAT. We select CCAT as a positive category in our experiment, and the rest of the documents are taken as negative samples, so as to generate a balanced data stream.

Usually, there are thousands of features in a corpus. We performed feature selection by using WEKA in the experi-

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

ment. As RCV1-V2 is too large for WEKA to process, for CCAT category, 3% of the training samples and 3% of the testing samples are selected randomly to form dataset D, and 100 top expressive features are selected by performing Information Gain (IG) [16] algorithm on D. The original text corpus is represented by these features.

The details of the dataset after the preprocessing are follows.

Table 3: Description of the Real Dataset.

Dataset	#Att	Training Set		Testing Set	
		#Pos	#Neg	#Pos	#Neg
RCV1-V2	101	370541	410724	10786	12363

4.2.2 Experiment Result

We simulate data stream on this real-life dataset. For RCV1-V2 dataset, *UnLevel* ranges from 65% to 80%. For each experiment setting, we generate fifty trails of data streams, and the averaged experiment results of the fifty trails are reported in Tab.4.

In Tab.4, column 1 lists the name of the dataset; columns 2 to 6 give the F1 index, accuracy index, running time, count of leaves, and count of inner nodes of VFDT respectively; column 7 gives the *UnLevel* parameter; and columns 8 to 12 give the experiment results for OcVFDT.

It is obvious from Tab.4 that even with high percentage of unlabeled samples, the performance of OcVFDT is very competitive to that of VFDT, which is trained on fully labeled data stream. Furthermore, with different *UnLevel* values, the performance of OcVFDT is stable.

5. CONCLUSION AND FUTURE WORK

In this paper, based on VFDT and POSC4.5, we propose OcVFDT algorithm for one-class classification of data streams. Our experiments on both synthetic data and real-life data show that even 80% of the samples in the stream are unlabeled, the classification performance of OcVFDT is still very close to that of VFDT which is trained on fully labeled data stream. For example, in synthetic datasets with 80% unlabeled samples, the decrease accuracy of OcVFDT of $|S| = 10^5, 10^6$ and 10^7 with $PosLevel = 50\%$ is 0.022, 0.008 and 0.009 respectively.

OcVFDT requires less labeled training samples, which makes it more applicable to real-life applications.

Data streams are characterized by concept drift. However, concept drift is not considered in this paper for simplicity. Based on our OcVFDT algorithm, we plan to study one-class data stream classification algorithm to cope with concept drift as our next effort.

6. REFERENCES

[1] B. Calvo, P. Larranaga, and J. A. Lozano. Learning Bayesian classifiers from positive and unlabeled examples. *Pattern Recognition Letters*, 28:2375–2384, 2007.

[2] F. Denis, R. Gilleron, and F. Letouzey. Learning from positive and unlabeled examples. *Theoretical Computer Science*, pages 70–83, 2005.

[3] T. Dietterich. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

[4] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (SIGKDD'00)*, pages 71–80. ACM New York, NY, USA, 2000.

[5] C. Elkan and K. Noto. Learning classifiers from only positive and unlabeled data. In *Proceedings of the Fourteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (SIGKDD'08)*.

[6] U. M. Fayyad and K. B. Irani. On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, 8:87–102, 1992.

[7] G. Fung, J. Yu, H. Lu, and P. Yu. Text classification without negative examples revisit. *IEEE Transactions on Knowledge and Data Engineering*, 18(1):6–20, 2006.

[8] J. Gama, P. Medas, and P. Rodrigues. Learning Decision Trees from Dynamic Data Streams. *Journal of Universal Computer Science*, 11(8):1353–1366, 2005.

[9] J. Gama, R. Rocha, and P. Medas. Accurate decision trees for mining high-speed data streams. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (SIGKDD'03)*, pages 523–528. ACM Press New York, NY, USA, 2003.

[10] G. Hulten, P. Domingos, and L. Spencer. Mining massive data streams. In *The Journal of Machine Learning Research*, 2005.

[11] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (SIGKDD'01)*, pages 97–106. ACM New York, NY, USA, 2001.

[12] R. Jin and G. Agrawal. Efficient decision tree construction on streaming data. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (SIGKDD'03)*, pages 571–576. ACM New York, NY, USA, 2003.

[13] W. Lee and B. Liu. Learning with Positive and Unlabeled Examples Using Weighted Logistic Regression. In *Proceedings of Twentieth International Conference on Machine Learning. (ICML'03)*, volume 20, page 448, 2003.

[14] D. Lewis, Y. Yang, T. Rose, and F. Li. RCV1: A New Benchmark Collection for Text Categorization Research. *The Journal of Machine Learning Research*, 5:361–397, 2004.

[15] B. Liu, Y. Dai, X. Li, W. Lee, and P. Yu. Building text classifiers using positive and unlabeled examples. In *Proceedings of the Third IEEE International Conference on Data Mining. (ICDM'03)*, pages 179–186, 2003.

[16] J. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[17] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. Smola, and R. Williamson. Estimating the Support of a

Table 4: Result of Experiments on Real Dataset.

Dataset	VFDT					OcVFDT					
	F1	Acc	Time(ms)	#Leaves	#Node	*UnLevel*	F1	Acc	Time(ms)	#Leaves	#Node
RCV1-V2	0.812	0.837	4550.05	184	183	80	0.801	0.830	49152.78	199.30	198.30
						75	0.802	0.830	49144.54	196.52	195.52
						70	0.802	0.827	50019.94	199.92	198.92
						65	0.795	0.827	50009.84	197.76	196.76

High-Dimensional Distribution. *Neural Computation*, 13(7):1443–1471, 2001.

[18] W. Street and Y. Kim. A streaming ensemble algorithm (SEA) for large-scale classification. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (SIGKDD'01)*, pages 377–382. ACM New York, NY, USA, 2001.

[19] H. Wang, W. Fan, P. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (SIGKDD'03)*, pages 226–235. ACM New York, NY, USA, 2003.

[20] H. Yu. Single-Class Classification with Mapping Convergence. *Machine Learning*, 61(1):49–69, 2005.

[21] H. Yu, J. Han, and K. Chang. PEBL: web page classification without negative examples. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):70–81, 2004.

[22] Y. Zhang and X. Jin. An automatic construction and organization strategy for ensemble learning on data streams. *ACM SIGMOD Record*, 35(3):28–33, 2006.

[23] Y. Zhang, X. Li, and M. Orlowska. One-Class Classification of Text Streams with Concept Drift. In *Proceedings of the Third IEEE International Conference on Data Mining Workshops. (ICDMW'08)*, pages 116–125, 2008.

[24] X. Zhu, X. Wu, and Y. Yang. Dynamic Classifier Selection for Effective Mining from Noisy Data Streams. In *Proceedings of the Fourth IEEE International Conference on Data Mining. (ICDM'04)*, pages 305–312, 2004.