Information Sciences 213 (2012) 50–67

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Learning very fast decision tree from uncertain data streams with positive and unlabeled samples

Chunquan Liang [a], Yang Zhang [b,c,*], Peng Shi [d,e], Zhengguo Hu [a]

[a] College of Mechanical and Electronic Engineering, Northwest A&F Univ., Shaanxi, China
[b] College of Information Engineering, Northwest A&F Univ., Shaanxi, China
[c] State Key Laboratory for Novel Software Technology, Nanjin Univ., Nanjin, China
[d] Dept. of Computing and Mathematical Sciences, Univ. of Glamorgan, Pontypridd, UK
[e] Sch. of Engineering and Science, Victoria Univ., Melbourne, Vic., Australia

ARTICLE INFO

Article history:
Received 23 August 2011
Received in revised form 23 May 2012
Accepted 25 May 2012
Available online 4 June 2012

Keywords:
Uncertain data stream
Very fast decision tree
Positive unlabeled learning
Uncertain attribute

ABSTRACT

Most data stream classification algorithms need to supply input with a large amount of precisely labeled data. However, in many data stream applications, streaming data contains inherent uncertainty, and labeled samples are difficult to be collected, while abundant data are unlabeled. In this paper, we focus on classifying uncertain data streams with only positive and unlabeled samples available. Based on concept-adapting very fast decision tree (CVFDT) algorithm, we propose an algorithm namely puuCVFDT (CVFDT for positive and unlabeled uncertain data). Experimental results on both synthetic and real-life datasets demonstrate the strong ability and efficiency of puuCVFDT to handle concept drift with uncertainty under positive and unlabeled learning scenario. Even when 90% of the samples in the stream are unlabeled, the classification performance of the proposed algorithm is still compared to that of CVFDT, which is learned from fully labeled data without uncertainty.

1. Introduction

Data stream classification has been widely used in many applications, e.g., credit fraud detection, network intrusion detection, and environmental monitoring. In these applications, tremendous amount of streaming data is collected at an unprecedented rate. In addition, this streaming data is normally characterized by its drifting concepts [15,32]. Thus, the major obstacles in data stream classification lie in memory, time, space, and evolving concepts. To address these obstacles, many algorithms have been proposed, mainly including various algorithms based on ensemble approach [30,32] and decision tree, e.g., [8,15,20] proposed algorithms to learn very fast decision trees, while [29] presented fuzzy pattern tree algorithm for binary classification on data streams. Among these algorithms, concept-adapting very fast decision tree (CVFDT) [15] is a well-known algorithm for data stream classification. By keeping its model consistent with a sliding window of samples and using Hoeffding bound theory [14], CVFDT can learn decision trees incrementally with bounded memory usage, high processing speed, and detecting evolving concepts.

To learn a classifier, the CVFDT algorithm requires input of precise and fully labeled samples, which is impractical in many real world applications. The streaming data often contains uncertainty due to various reasons, such as imprecise measurement, missing values, and privacy protection. In addition, labeled samples are expensive to collect while

* Corresponding author at: College of Information Engineering, Northwest A&F Univ., Shaanxi, China.
E-mail address: zhangyang@nwsuaf.edu.cn (Y. Zhang).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2012.05.023

unlabeled data may be abundant. Meanwhile for some binary classification problems, the target concept elements (we call them positive samples in this paper) are obtained easily [6]. These situations are predominant in data stream application. For example, in credit fraud detection, firstly, private information of customers such as ages, addresses and vocations may be masked by imprecise values when published for mining propose. Secondly, if customer behaviors have not caused any bad effect, we need to investigate thoroughly to decide whether they are fraud or not. It would be extremely expensive to label a huge volume of these customers. So it is better to use them as unlabeled data. Finally, this application could be modeled as a binary classification problem, while those behaviors which lead to bad effect could be labeled as positive samples. The same scenario could also be observed in environmental monitoring, network intrusion detection, and so on. Thus, it is helpful to design novel classification algorithms for these uncertain data streams with positive and unlabeled samples.

To the best of our knowledge, the problem of classifying uncertain data streams with only positive and unlabeled samples has not been studied by the research community yet. In this paper, based on the recent works on building uncertain decision trees [23,31] and learning from positive and unlabeled samples [6], we address the problem of learning very fast decision trees by using only positive and unlabeled samples with both certain and uncertain attributes. We transform CVFDT [15] to cope with both numerical and categorical data with uncertainty under positive and unlabeled learning (PU learning) scenario, and propose a novel algorithm namely puuCVFDT (CVFDT for Positive and Unlabeled samples with Uncertainty). A series of experiments on both synthetic and real-life dataset show that the proposed puuCVFDT algorithm has strong capabilities to learn from uncertain data streams with positive and unlabeled samples and tackle concept drift. Even when only 10% of the samples in the stream are positive, the classification accuracy of puuCVFDT is still very close to that of CVFDT, which is trained on fully labeled data stream without uncertainty.

In the rest of this paper, we firstly discuss related works in Section 2. Problem definition is described in Section 3. Our method to cope with uncertain data streams under PU learning scenario is discussed in Section 4 and Section 5. Algorithm details are given in Section 6. The experimental study is presented in Section 7, and we conclude our paper in Section 8.

2. Related works

2.1. Positive and unlabeled learning

The goal of PU learning is to learn a classifier with only positive and unlabeled data. Many data mining algorithms, such as decision tree algorithms and naive Bayes algorithms, could be considered as statistical query (SQ-like) learning algorithms, since they only use samples to evaluate statistical queries [6]. In [6], Denis et al. have presented a scheme to convert any SQ-like learning algorithm into positive and unlabeled learning algorithm. They have also given that it is possible to learn from positive and unlabeled samples if the weight of the target concept is available to the learner. Then they proposed POSC4.5, a decision tree learning algorithm, to learn from positive and unlabeled samples. Other works on PU learning mainly focus on: (1) Approaches to transforming PU learning problems into traditional problems by extracting negative samples from unlabeled ones [10,19,35]; (2) Approaches to building basic classifiers. For example, [28] proposed a SVM-based PU learning algorithm in [28]. Assuming that labeled samples were chosen randomly from the positive samples, [9] gave two methods for PU learning in [9].

2.2. Semi-supervised learning

Unlabeled data has also been well explored in plenty of semi-supervised learning (SSL) methods. By using large amount of unlabeled data, SSL tries to learn better classifiers than those trained by purely supervised learning methods. For several years, there have been many SSL methods proposed, mainly including self-training, co-training, multi-view learning, transductive Support Vector Machine (SVM), and graph-based methods [4]. For example, [37] presented a new generative model for multi-view learning via probabilistic latent semantic analysis. [22] introduced a transductive SVM classification tree for personalized transductive learning. By using unlabeled data, SSL requires less human effort but achieves higher accuracy. So it is of great interest in many applications, such as natural language problems [18], web page ranking [11] and image/video annotation [33]. A detailed introduction to SSL could be found in [4], and extensive reviews of the existing methods have been provided in [36]. Although unlabeled data is used in both SSL and PU learning, there is a great difference between them. SSL is designed to learn classifications models from both labeled and unlabeled samples. While the unlabeled ones are exploited to improve accuracy of supervised learning algorithms. PU learning however is designed to learn binary classifiers with only positive and unlabeled samples, even though there are no negative ones. While the unlabeled ones are exploited to estimate negative class conditional probability. In this paper, we design an algorithm under PU learning scenario.

2.3. Classification of uncertain data

Uncertain data classification is an important branch of uncertain data mining [5,7]. There have been several uncertain classification algorithms proposed by extending traditional models to cope with uncertainty. Researches in [23,31] proposed uncertain decision tree algorithms DTU and UDT respectively. Considering the uncertain value as multiple values forming a

probability distribution function, they adopt the idea of fractional sample (in this paper we also use the technique of fractional sample to handle uncertainty) to assign an uncertain sample into a tree. At each node, uncertain samples are used to estimate probabilistic statistics, and these statistics are used to evaluate a heuristic measure, such as information gain, for building the tree. Gao and Wang [12], Qin et al. [24] present the rule-based classification algorithms for uncertain data. In [2], SVM is extended to TSVM to classify data with uncertainty caused by noisy. He et al. [13], Qin et al. [25], and Ren et al. [27] address Naive Bayes approach to uncertain data classification problem. These algorithms are all designed for static dataset. While this work will learn classifiers from streaming data with uncertainty.

2.4. Classification of data streams with uncertainty or PU leaning scenario

Most of works on data stream classification are devoted to supervised learning without uncertainty [30,32,8,15,20], while only a few works focus on data streams with uncertainty or PU learning scenario. For example, Pan et al. [21] proposes to handle label uncertainty in data stream based on ensemble algorithm. Li et al. [16] proposes one-class very fast decision tree (OcVFDT) algorithm to learn data stream using positive and unlabeled samples. However, the OcVFDT algorithm can only cope with certain categorical attributes, and concept drift is not considered. In our previous work [17], we have given method to handle uncertain categorical data in CVFDT, and proposed uncertainty-handling and concept-adapting very fast decision tree (UCVFDT) algorithm. However, the UCVFDT algorithm needs to supply input with fully labeled data, and it cannot handle numerical uncertainty.

3. Problem definition

3.1. Data model

Uncertainty could arise in both numerical and categorical data. An attribute with imprecise data is called uncertain attribute. We write $\mathbb{X}^u$ for the set of uncertain attributes, and write X_i^u for the ith attribute in $\mathbb{X}^u$. An attribute X_i^u can be an uncertain numerical attribute (UNA) or uncertain categorical attribute (UCA). Based on [23,31], we define UNA and UCA as follows.

Definition 1 ([23,31]). We write X_i^{un} for the ith UNA, and write X_{it}^{un} for the value of X_i^{un} on the tth sample s_t. Then X_{it}^{un} is defined as a probability distribution function (*pdf*) $f_{it}(x)$ over a range $[a_{it}, b_{it}]$, where $\int_{b_{it}}^{a_{it}} f_{it}(x)dx = 1$. Thus, the value of attribute X_i^{un} of sample s_t is denoted as $X_{it}^{un} = \{f_{it}(x)|x \in [a_{it}, b_{it}]\}$.

A certain numerical attribute X_i with value v_{it} can be modeled as a special case of X_i^{un} with $f_{it}(x) = 1$ and $a_{it} = b_{it} = v_{it}$. For simplicity, all numerical attributes will be modeled as uncertain ones in the rest of this paper.

Definition 2 [23]. We write X_i^{uc} for the ith UCA, and write X_{it}^{uc} for the value of X_i^{uc} on the tth sample s_t. Then X_{it}^{uc} is characterized by a *pdf* over a categorical domain $Dom(X_i^{uc}) = \{v_1, \ldots, v_m\}$. It can be represented by the probability vector $P = \langle p_{i1}, \ldots, p_{im}\rangle$ such that $P(X_{it}^{uc} = v_j) = p_{ij}$ and $\sum_{j=1}^{m} p_{ij} = 1$.

A certain categorical attribute X_i with value v_j can be modeled as a special case of X_i^{uc} with $P(X_{it}^{uc} = v_j) = 1$ and $P(X_{it}^{uc} = v) = 0$ for all $v \neq v_j$. For simplicity, all categorical attributes will be modeled as uncertain ones in the rest of this paper.

Definition 3. Uncertain data stream with positive and unlabeled samples could be modeled as $S = \langle s_1, s_2, \ldots, s_t, \ldots\rangle$, where s_t is the most up-to-date sample in the stream. Under the PU learning scenario, a sample s_t from the stream is represented by $s_t = \langle \mathbb{X}^u, y, l\rangle$, where $\mathbb{X}^u = \{X_1^u, X_2^u, \ldots, X_d^u\}$ is a tuple with d attributes X_i^u and each attribute could be a UNA (X_i^{un}) or a UCA (X_i^{uc}). $y \in \{y_0, y_1\}$ represents the class label of sample s_t, with y_0 for negative and y_1 for positive. $l \in \{l_0, l_1\}$ represents whether the class label of s_t is available to the learner (l_1) or not (l_0). Thus, a positive sample could be denoted as $s_t = \langle \mathbb{X}^u, y_1, l_1\rangle$, while an unlabeled sample could be denoted as $s_t = \langle \mathbb{X}^u, ?, l_0\rangle$, where ? represents that the class label of s_t is not available for the learner.

3.2. CVFDT for data stream classification

To learn a CVFDT tree, a leaf is recursively replaced with a decision node, starting from the root. Each leaf does not store the incoming samples but some sufficient statistics about attribute-values. These statistics are used to evaluate the heuristic measure for choosing the best split attribute. The incoming sample is assigned recursively to tree nodes, and corresponding sufficient statistics are incrementally updated. To decide how many samples are necessary to choose a split test at each leaf, a statistical result known as Hoeffding bound [14] is used.

Suppose r is a real-valued random variable with range R. If we have made n independent observations of r, then with probability $1 - \delta$, the Hoeffding bound guarantees that the difference between the true mean and observed mean is less than ε, where $\varepsilon = \sqrt{R^2 ln(1/\delta)/2n}$ [15]. Let $G(X_i)$ be the evaluation function used to choose test attributes. For information gain, the

range R of $G(X_i)$ is $\log_2 C$, where C is the number of classes. Let X_a and X_b be the attribute with the highest and second-highest observed $\overline{G}$ respectively, and $\Delta\overline{G} = \overline{G}(X_a) - \overline{G}(X_b)$ [15]. Given a desired δ, after observing n samples, if we have $\Delta\overline{G} > \varepsilon$, then the Hoeffding bound ensures that X_a is the correct choice with probability $1 - \delta$ [15].

3.3. Classifying uncertain data stream with positive and unlabeled samples

Given an uncertain stream data $S = \langle s_1, s_2, \ldots, s_t, \ldots\rangle$, our goal is to build a fast uncertain decision tree model $y = f(\mathbb{X}^u)$ from S and classify the test samples into positive or negative. We summarize three following problems in this paper:

Problem 1. In CVFDT, the incoming sample will be passed down to a certain node from the root recursively according to its attribute value. However, under uncertainty model, an uncertain attribute value is represented not by a precise value, but by a *pdf* over a range. The problem here is how to assign an incoming uncertain sample to tree nodes. We will discuss this problem in Section 4.

Problem 2. Data streams are characterized by huge volume of endless data arriving at high speed, and hence it is unrealistic to store all samples for multiple scans when building the decision tree. Furthermore, splitting measures, such as information gain [26] and the Gini index [3], are not applicable to positive and unlabeled samples with uncertainty. So the problem here is the way to maintain sufficient statistics for incoming stream data, the way to define the splitting measure for positive and unlabeled samples with both uncertain numerical data and uncertain categorical data, and the way to use the sufficient statistics to evaluate this splitting measure. We will cope with these problems in Section 5.1, Section 5.2, and Section 5.3.

Problem 3. How to calculate the Hoeffding bound for uncertain data streams? We will solve this problem in Section 5.4.

After tackling the above problems, we will give the details of our puuCVFDT algorithm in Section 6.

The notations used in this paper is summarized in Table 1.

4. Fractional sample

We adopt the idea of fractional sample [26,31] to assign an uncertain sample to tree nodes. For a sample $s_t = \langle \mathbb{X}^u, y, l\rangle$ observed by internal node N, the following sections introduce how to split it into fractional samples and how to assign these fractional samples to the child of node N.

4.1. Split on uncertain numerical attribute

Let $X_i^{un} \in \mathbb{X}^u$ be the split attribute at node N, and z_i be the split point. For simplicity, we use binary split in this paper. Then attribute X_i^{un} splits the uncertain interval $[a_{it}, b_{it}]$ of s_t into 2 sub-intervals at point z_i. We compute the probability

Table 1
Summary of notations

Notation	Description
$\mathbb{X}^u$	Set of uncertain attributes
X_i^u	The ith uncertain attribute in $\mathbb{X}^u$
X_i^{un}/X_i^{uc}	The ith uncertain numerical/categorical attribute
X_{it}^{un}/X_{it}^{uc}	Value of X_i^{un}/X_i^{uc} on the tth sample
y	Class label $y \in \{y_0, y_1\}$, with y_0 for negative, y_1 for positive
l	Whether the class label is available to the learner, $l \in \{yes(l_1), no(l_0)\}$
s_t, w_t	The tth sample and its weight, $s_t = \langle \mathbb{X}^u, y, l\rangle$
S	Uncertain data stream with positive and unlabeled samples
p_1/p_0	Ratio of positive/negative samples in current data stream S
$f_{it}(x)$	Probability distribution function of X_{it}^{un}
$[a_{it}, b_{it}]$	Range of X_{it}^{un}
z_i	Split point at attribute X_i^{un}
$Dom(X_i^{uc})$	Categorical domain of X_i^{uc}
$PC(S)$	Probabilistic cardinality of samples in S
$G(X_i)$	Evaluation function used to choose test attributes.
$puulG(S, X_i^u)$	Uncertain information gain for S that splitting on X_i^u

$p_{i1} = \int_{a_{it}}^{z_i} f_{it}(x)dx$ (if $z_i < a_{it}$ then $p_{i1} = 0$), and $p_{i2} = \int_{z_i}^{b_{it}} f_{it}(x)dx$ (if $z_i > b_{it}$ then $p_{i2} = 0$) [31]. Thus, s_t will be split into 2 fractional samples, s_1 and s_2. These two samples will be passed down to the left and the right child of node N respectively [31].

We associate s_t with weight w_t, which can be interpreted as the existing probability of s_t at node N. The weight of s_1 is assigned with $w_1 = w_t * p_{i1}$, which can be interpreted as the probability that s_t falls into the left child. Then, the *pdf* of s_1 for X_i^{un} is given by $f_1(x) = f_{it}(x)/w_1, x \in [a_{it}, z_i]$, and $f_1(x) = 0$, if $x \notin [a_{it}, z_i]$ [31]. Hence, $X_{it_1}^{un} = \{f_1(x)|x \in [a_{it}, z_i]\}$. Thus, we have $s_1 = \langle \mathbb{X}^u \cup X_{it_1}^{un} - X_{it}^{un}, y, l\rangle$ and s_2 is generated analogously.

A special case is that X_{it}^{un} is a certain attribute and has a point value v_{it}. In this case, if $v_{it} <= z_i, s_t$ will pass down to the left child of node N, otherwise, to the right child.

4.2. Split on uncertain categorical attribute

Let $X_i^{uc} \in \mathbb{X}^u$ be the split attribute at node N. It splits s_t into m samples $\{s_{t1}, s_{t2}, \ldots, s_{tm}\}$, where $m = |Dom(X_i^{uc})|$, and s_{tj} is a copy of s_t except for attribute X_i^{uc}, which is set to $P(X_{ij}^{uc} = v_j) = 1$, and for all $v \neq v_j, P(X_{ij}^{uc} = v) = 0$ [31]. Thus s_{tj} is a fractional sample of s_t splitting on attribute X_i^{uc} and it will be assigned to jth child of node N. The weight of s_{tj} is assigned with $w_{tj} = w_t^* P(X_{it}^{uc} = v_j)$, which can be interpreted as the probability that s_t falls into the jth child of node N.

A special case is that X_{it}^{uc} has a certain value v_j, that is, $P(X_{it}^{uc} = v_j) = 1$. In this case, s_t will completely falls into the jth child of node N.

When growing a puuCVFDT tree, an uncertain sample is partitioned into fractional samples and assigned to tree nodes, starting from the tree root recursively. The sufficient statistics at each node are collected from these fractional samples. And these statistics will be used to evaluate the heuristic measure for identifying the best split attribute.

5. Choosing the splitting attribute

To choose the splitting attribute is the key issue of building a fast decision tree. At each step of tree growing, splitting measures are evaluated for selecting the test attribute. In this section, we firstly define uncertain information gain for positive and unlabeled samples (puuIG) as splitting measures. Then we give the way to maintain the sufficient statistics incrementally for both UNA and UCA, and the way to use these statistics to evaluate puuIG.

5.1. Uncertain information gain for positive and unlabeled samples

Let $S_{\mathcal{R}}$ be the set of samples observed at the tree root $\mathcal{R}$, S_N be the set of samples observed at node N, and *PosLevel* be the probability to observe target concept in current data stream. Suppose attribute X_i^u (numerical or categorical) is selected as the split attribute, and it splits set S_N into m subsets of fractional samples, $\{S_{i1}, S_{i2}, \ldots, S_{im}\}$. We define uncertain information gain for positive and unlabeled samples below:

$$puuIG(S_N, X_i^u) = puuEntropy(S_N) - \sum_{j=1}^{m} \frac{PC(S_{ij})}{PC(S_N)} \times puuEntropy(S_{ij}), \tag{1}$$

where $PC(S) = \sum_{s_t \in S} w_t$ denotes the probabilistic cardinality [23] of samples in S, and $puuEntropy(S)$ denotes expected information entropy, which is defined as:

$$puuEntropy(S) = -p_1 \log_2 p_1 - p_0 \log_2 p_0, \tag{2}$$

where p_k is the estimated ratio of positive samples ($k = 1$) or negative samples ($k = 0$) in current data stream S. Based on [6], p_1 and p_0 can be estimated by:

$$p_1 = min\left(\frac{PC(S, l_1)}{PC(S_{\mathcal{R}}, l_1)} \times PosLevel \times \frac{PC(S_{\mathcal{R}}, l_0)}{PC(S, l_0)}, 1\right), p_0 = 1 - p_1, \tag{3}$$

where $PC(S, l_k) = \sum_{s_t \in S} w_t \times P(l(s_t) = l_k)$ is the probabilistic cardinality of positive samples ($k = 1$) or unlabeled samples ($k = 0$) in S. As we can see, the key to evaluate puuIG is to determine probabilistic cardinalities $PC(S_N), PC(S_N, l_k), PC(S_{\mathcal{R}}, l_k), PC(S_{ij})$ and $PC(S_{ij}, l_k)$.

5.2. Uncertain numerical attribute

Suppose attribute X_i^{un} and z_i is selected as the split attribute and split point respectively, then it splits set S_N into 2 subsets of fractional samples. The cardinalities over these two subsets are computed directly below:

$$PC(S_{i1}, l_k) = \sum_{s_t \in S_{i1}} w_t \times P(l(s_t) = l_k) = \sum_{s_t \in S_N} w_t \times \int_{a_{it}}^{z_i} f_{it}(x)dx \times P(l(s_t) = l_k) \quad (4)$$

$$PC(S_{i2}, l_k) = \sum_{s_t \in S_{i2}} w_t \times P(l(s_t) = l_k) = \sum_{s_t \in S_N} w_t \times \int_{z_i}^{b_{it}} f_{it}(x)dx \times P(l(s_t) = l_k) \quad (5)$$

$$PC(S_{i1}) = \sum_{s_t \in S_{i1}} w_t = \sum_{s_t \in S_N} w_t \times \int_{a_{it}}^{z_i} f_{it}(x)dx \quad (6)$$

$$PC(S_{i2}) = \sum_{s_t \in S_{i2}} w_t = \sum_{s_t \in S_N} w_t \times \int_{z_i}^{b_{it}} f_{it}(x)dx \quad (7)$$

Obviously, to determine the cardinalities following the formula (4)–(6) and formula (7), multiple passes over the set S_N are necessary. However, for the sake of efficiency and limited memory, the method of multiple scans is not applicable to data stream scenarios. In next subsection, we collect some sufficient statistics by making a single pass over the set S_N, and show how to use these sufficient statistics to compute puuIG efficiently.

5.2.1. Gaussian approximation

Berbgard et al. proposed to use Gaussian approximation (GA) to approximate the distribution of precise numerical values in [1]. Here, we extend this idea to uncertain data by using Gaussian distribution function $F(\overline{X}_{ik}^{un}, \sigma_{ik}^2, x)$ to approximate uncertain attribute X_i^{un} of positive samples ($k = 1$) and unlabeled samples ($k = 0$). This distribution can be maintained incrementally by storing only three numbers in memory, the mean $\overline{X}_{ik}^{un}$, the variance σ_{ik}^2, and sum of weight Σ_{ik}. These sufficient statistics, $(\overline{X}_{ik}^{un}, \sigma_{ik}^2, \Sigma_{ik})$, will be used to compute puuIG for building the puuCVFDT tree. Let s_t with weight w_t be the fractional sample observed by current node N, that is, $s_t \in S_N$. Based on the work [34], our Gaussian approximation method is described in detail below.

A. Initialization. For a new leaf N and for each uncertain attribute $X_i^{un} \in \mathbb{X}^u$ with l_k, $(\overline{X}_{ik}^{un}, \sigma_{ik}^2, \Sigma_{ik})$ is set to $(0, 0, 0)$.

B. Discretization. The uncertain interval (a_{it}, b_{it}) of s_t is divided into M uniform cells. Let $\Delta = 1/M$, then $f_{it}(x)$ can be approximated by $f_{it}(n\Delta)$, where $n \in I_M = \{0, 1, \ldots, M\}$, given a M. Thus, an uncertain numerical attribute is approximated by a sequence of value-weight pair (v_n, w_n). Here, $v_n = n\Delta$ and $w_n = w_t \Delta f_{it}(n\Delta)$. For the case of point value v_{it} of X_{it}^{un}, we have $n = 1$, $v_n = v_{it}$, and $w_n = w_t$.

C. Inserting a new sample to ***N***. When a new sample reaches N, for each (v_n, w_n), the sufficient statistics $(\overline{X}_{ik}^{un}, \sigma_{ik}^2, \Sigma_{ik})$ is updated by:

$$\Sigma_{ik} = \Sigma_{ik} + w_n.$$
$$Temp = \overline{X}_{ik}^{u}.$$
$$\overline{X}_{ik}^{un} = \overline{X}_{ik}^{un} + w_n(v_n - Temp)/\Sigma_{ik}.$$
$$\sigma_{ik}^2 = (\Sigma_{ik} - w_n - 1)\sigma_{ik}^2 + w_n(v_n - Temp)(v_n - \overline{X}_{ik}^{un})/(\Sigma_{ik} - 1),$$

where if $\Sigma_{ik} \leqslant 1$, then $\sigma_{ik}^2 = 0$.

D. Deleting an outdated sample from ***N***. To track concept drift, our puuCVFDT keeps its model consistent with a sliding window of samples. A sample becomes outdated when it leaves the window. As an outdated sample cannot represent the current concept any more, it should be deleted from N. For each (v_n, w_n), the sufficient statistics $(\overline{X}_{ik}^{un}, \sigma_{ik}^2, \Sigma_{ik})$ is updated by:

$$\Sigma_{ik} = \Sigma_{ik} - w_n.$$
$$Temp = \overline{X}_{ik}^{un} - w_n(v_n - \overline{X}_{ik}^{un})/\Sigma_{ik}.$$
$$\sigma_{ik}^2 = (\Sigma_{ik} + w_n - 1)\sigma_{ik}^2 - w_n(v_n - Temp)(v_n - \overline{X}_{ik}^{un})/(\Sigma_{ik} - 1).$$
$$\overline{X}_{ik}^{un} = Temp.$$

Here, if $\Sigma_{ik} \leqslant 1$, then $\sigma_{ik}^2 = 0$.

E. Calculating puuIG. Let $V_{ik}^{min} = \overline{X}_{ik}^{un} - \mu \times \sigma_{ik}^2$ and $V_{ik}^{max} = \overline{X}_{ik}^{un} + \mu \times \sigma_{ik}^2$, where μ is a user specified confidence threshold (see Section 5.2.2 for details of this parameter). Then, the splitting point z_i is determined inside the interval $(minvalue, maxvalue)$. Here, $minvalue = \min(V_{ik}^{min})$ and $maxvalue = \max(V_{ik}^{max})$ where $k \in \{0, 1\}$. The interval $(minvalue, maxvalue)$ is divided into Z uniform cells. Let $\Delta = 1/Z$, for uncertain attribute X_{un}^i and for each $z_i = n\Delta$, where $n \in I_Z = \{1, 2 \ldots, Z\}$, given Z, we have:

$$PC(S_{i1}, l_k) = \Sigma_{ik} F(\overline{X}_{ik}^{u}, \sigma_{ik}^2, z_i)$$
$$PC(S_{i2}, l_k) = \Sigma_{ik} - PC(S_{i1}, l_k)$$
$$PC(S_N, l_k) = \Sigma_{ik}$$
$$PC(S_N) = \sum_{k \in \{0,1\}} PC(S_N, l_k) = \Sigma_{i0} + \Sigma_{i1}$$

The probabilistic cardinalities over $S_{\mathcal{R}}$ is calculated analogously. Thus, the puuIG index of attribute X_i^{un} can be calculated by substituting the above equation into formula (1), formula (2), and formula (3).

5.2.2. Performance analysis

Confidence threshold parameter μ. Here, we show how the parameter μ affect the interval for determining the position of split point z_i. According to the characteristics of Gaussian distribution, given $\mu = 1.96$, we have $\alpha = P(V_{ik}^{min} < x < V_{ik}^{max}) > 95\%$. Then, we also have

$$\begin{aligned} &P(minvalue < x < maxvalue) \\ &\quad = P\left(\min_{k\in\{0,1\}}(V_{ik}^{min}) < x < \max_{k\in\{0,1\}}(V_{ik}^{max})\right) \\ &\quad \geqslant \max_{k\in\{0,1\}}(P(V_{ik}^{min} < x < V_{ik}^{max})) > 95\%. \end{aligned}$$

This inequation shows that, for a given $\mu = 1.96$, the best z_i locates in $(minvalue, maxvalue)$ with probability 95%. We will give experimental study on parameter μ in Section 7.2.4.

Time complexity. $F(\overline{X}_{ik}^{un}, \sigma_{ik}^2, x)$ is a Gaussian distribution function which can be obtained in a small constant time. To determine the best puuIG index, the time complexity for the proposed Gaussian approximation and the multiple scan method are $O(Z * d)$ and $O(Z * d * nSample)$ respectively, where Z is the number of split points to evaluate, d is the number of attributes, and $nSample$ is the number of fractional samples observed at node N.

Space complexity. To store the sufficient statistic for samples observed by node N, the space complexity for the proposed Gaussian approximation and the multiple scan method are $O(d)$ and $O(d * nSample)$ respectively.

As discussed above, compared with the multiple scan method, the proposed Gaussian approximation method works with better time complexity and space complexity.

5.3. Uncertain categorical attribute

Suppose attribute X_i^{uc} is selected as the split attribute, then it splits set S_N into $m = |Dom(X_i^{uc})|$ subsets of fractional samples. To evaluate puuIG, the difference from attribute X_i^{un} lies in calculating the probabilistic cardinality over S_{ij}, which is given by:

$$PC(S_{ij}) = \sum_{s_t \in S_N} w_t \times P(X_{it}^{uc} = v_j) \tag{8}$$

$$PC(S_{ij}, l_k) = \sum_{s_t \in S_N} w_t \times P(X_{it}^{uc} = v_j) \times P(l(s_t) = k) \tag{9}$$

To determine the cardinalities following formula (8) and formula (9), multiple passes over set S_N are also required. So they are not applicable to data stream scenarios. Instead, at each node of decision tree, for each possible value v_j of each attribute $X_i^{uc} \in \mathbb{X}^u$ in l_k, we associate it with a sufficient statistic n_{ijk}. In order to collect statistical data for puuIG, we only make a single pass over set S_N. For each new coming sample $s_t \in S_N$ with weight w_t, statistic n_{ijk} is updated below:

$$n_{ijk} = n_{ijk} + w_t \times P(l(s_t) = k) \times P(X_{it}^{uc} = v_j) \tag{10}$$

Similarly, to remove an outdated sample s_t from node N, statistic n_{ijk} is updated as follows:

$$n_{ijk} = n_{ijk} - w_t \times P(l(s_t) = k) \times P(X_{it}^{uc} = v_j) \tag{11}$$

The probabilistic cardinalities over the set S_N and S_{ij} can be calculated by $PC(S_{ij}) = \sum_{k\in 0,1} n_{ijk}, PC(S_{ij}, l_k) = n_{ijk}, PC(S_N) = \sum_{j=1}^{m} PC(S_{ij})$ and $PC(S_N, l_k) = \sum_{j=1}^{m} PC(S_{ij}, l_k)$. By substituting these equation into formula (1), formula (2) and formula (3), the puuIG index can be calculated.

5.4. Probabilistic Hoeffding bound

In Section 4 we note that, for uncertain data streams, the count for n observations at node N is expressed by expected count, which is obtained by $PC(S_N)$. So, we have $\varepsilon = \sqrt{R^2 ln(1/\delta)/2PC(S_N)}$. We name this Hoeffding bound for uncertain data stream as *probabilistic Hoeffding bound*.

6. The puuCVFDT algorithm

In this section, to build very fast decision tree from uncertain data streams with positive and unlabeled samples available, we give the details of our puuCVFDT algorithm.

Algorithm 1. Building uncertain very fast decision tree for positive and unlabeled samples

Input:

S a sequence of positive and unlabeled samples with uncertainty,
$\mathbb{X}^u$ a set of uncertain attributes,
$G(.)$ uncertain information gain for positive and unlabeled samples,
δ one minus the desired probability of choosing the correct attribute at any given node,
τ a user-supplied tie threshold,
w the size of the window,
n_{min} the number between checks for growth,
f the number between checks for drift,
ζ a user-supplied weight threshold,
$p_{validate}$ a user-supplied probability for selecting a sample as validate sample,
$n_{validate}$ the number between estimations for getting the best tree,

Output:

HT a decision tree for uncertain samples.

1: Let $T = \phi$.
2: For each $j \in [1, 9]$
3: Initialize a tree T_j with only a leaf $\mathcal{L}_j$ (the root).
4: Let $ALT(\mathcal{L}_j)$ be an initially empty set of alternate trees.
5: For $l_k \in \{l_0, l_1\}$ and each attribute $X_i^{un} \in \mathbb{X}^u$
6: Initialize sufficient statistics $(\overline{X}_{ik}^{un}, \sigma_{ik}^2, \Sigma_{ik})$ to $(0, 0, 0)$ at $\mathcal{L}_j$.
7: For $l_k \in \{l_0, l_1\}$ and each possible value v_j of each attribute $X_i^{uc} \in \mathbb{X}^u$
8: Initialize sufficient statistics $n_{ijk} = 0$ at $\mathcal{L}_j$.
9: $T = T \bigcup T_j$.
10: $T_{best} = T_1$.
11: For each sample $s_t \in S$
12: For each $j \in [1, 9]$
13: $PosLevel = j/10$.
14: $puuCVFDT(T_j, s_t, G, \delta, \tau$,w,$n_{min}, f, \zeta, PosLevel)$.
15: If $Random() \leqslant p_{validate}$ then
16: $Estimate(T, s_t)$.
17: If there have been $n_{validate}$ validate samples since the last getting the best tree
18: $T_{best} = GetBestTree(T)$.
19: return T_{best}.

6.1. Learning puuCVFDT

Based on [15,16], the process of building puuCVFDT is illustrated in Algorithm 1. In Algorithm 1, function $G(.)$ can be any function for choosing the split attribute. In this paper we use puuIG as $G(.)$; ζ is a threshold for weight of a sample; $p_{validate}$ is a user-supplied probability for selecting a sample as validate sample, while $n_{validate}$ is the number of samples observed between estimates for getting the best tree. Please refer to [15] for details of other parameters.

As the parameter *PosLevel*, the probability to observe the target concept is unknown, we follow the approach in [6,16] to estimate this parameter. Nine possible values, from 0.1 to 0.9, are enumerated as the value of *PosLevel*. Thus a forest *T*of nine puuCVFDT is built (Steps 1–14). Here, $ALT(\mathcal{L}_j)$ represents a set of alternate trees of node $\mathcal{L}_j$ (Step 4). Please refer to [15] for details of alternate trees. Then, with probability $p_{validate}$, the incoming sample s is used to estimate the classification performance of trees in T (Steps 15 and 16) by function *Estimate (T,s)*. The function *GetBestTree (T)* is used to get the best estimated tree from forest T. We will discuss these two functions in Section 6.2.

Algorithm 2 gives the details of building a single puuCVFDT, which follows the algorithm framework of CVFDT [15]. Firstly, like CVFDT with the ability of detecting concept drift, puuCVFDT works by keeping its model consistent with a sliding window of samples (Steps 1–3). Secondly, from Step 4 to Step 7, an outdated sample is removed from the tree and deleted from the sliding window (Algorithm 4). Thirdly, Step 8 is for tree growing (Algorithm 3). And finally, puuCVFDT checks split validity of an internal node periodically (Steps 9 and 10). If the chosen splitting attribute would no longer be selected, puuCVFDT grows alternate subtree and only modifies the decision tree when the alternate subtree is more accurate. Please refer to [15] for more details.

Algorithm 3 lists the pseudo code for growing uncertain decision tree. Sufficient statistics are collected from the incoming sample by the nodes it reaches (Steps 3–7). An internal node splits original samples into m fractional samples, and the subtrees of the current node start new growing processes with these fractional samples (Steps 10–15). At a leaf node, the puuIG index is computed using sufficient statistics that it collected. Then, the split attribute is chosen based on probabilistic Hoe-

ffding bound, and the leaf node is split into an internal node (Steps 17–31). The alternate trees of each node grow analogously (Steps 8 and 9).

Algorithm 2. Building a single puuCVFDT tree

puuCVFDT$(HT, s_t, G, \delta, \tau$,w$, n_{min}, f, \zeta, PosLevel)$
//Refer to Algorithm 1 for the details of input parameters.
1: Let W be the window of samples for HT.
2: Let ID be the maximum ID of the leaves that s_t can reach when split and sorted in HT.
3: Add (s_t, ID) to the beginning of W.
4: If $|W| >$w
5: Let (s_w, ID_w) be the last element of W.
6: *ForgetSample*$(HT, s_w, 1, ID_w, \zeta)$. //Algorithm 4.
7: Let $W = W$ with (s_w, ID_w) removed.
8: *puuCVFDTGrow*$(HT, G, s_t, 1, \delta, n_{min}, \tau, \zeta, PosLevel)$. //Algorithm 3.
9: If there have been f samples since the last checking of alternate trees
10: *CheckSplit (HT,* δ*)*

Algorithm 3. The process of growing a puuCVFDT tree

puuCVFDTGrow$(HT, G, \langle \mathbb{X}^u, y, l\rangle, w_t, \delta, n_{min}, \tau, \zeta, PosLevel)$
// Refer to Algorithm 1 for the details of input parameters.
1: If $w_t < \zeta$ then
2: Return.
3: Let N be the root of HT.
4: For each $l_k \in \{l_0, l_1\}$ and each attribute $X_i^{un} \in \mathbb{X}^u$
5: Update $(\overline{X}_{ik}^{un}, \sigma_{ik}^2, \Sigma_{ik})$ at N following GA. //Refer to Section 5.2.1
6: For $l_k \in \{l_0, l_1\}$ and each possible value v_j of each attribute $X_i^{uc} \in \mathbb{X}^u$
7: Update n_{ijk} at N following formula (10).
8: For each tree T_{ALT} in *ALT (N)*
9: *puuCVFDTGrow*$(T_{ALT}, G, \langle \mathbb{X}^u, y, l\rangle, w_t, \delta, n_{min}, \tau, \zeta, PosLevel)$.
10: If N is not a leaf, then
11: Split $(\mathbb{X}^u, y, l)$ into a set of m fractional samples E.
12: For each sample $s_j \in E$
13: Assign s_j with w$_j$. //Refer to Section 4
14: Let N_j be the branch child of N for s_j.
15: *puuCVFDTGrow*$(N_j, G, s_j, w_j, \delta, n_{min}, \tau, \zeta, PosLevel)$.
16: Else
17: Let n_1 and n_2 be the expect count of samples last seen and current seen at N.
18: If the samples seen so far at N are not all of the same class and $n_2 - n_1 > n_{min}$, then
19: Compute $\overline{G}(X_i^u)$ using $(\overline{X}_{ik}^{un}, \sigma_{ik}^2, \Sigma_{ik})$ or n_{ijk}. //Refer to Section 5.1
20: Let X_a^u and X_b^u be the attribute with highest and second-highest $\overline{G}$.
21: Let $\varepsilon = \sqrt{R^2 ln(1/\delta)/2PC(S_N)}$.
22: $\triangle\overline{G} = \overline{G}(X_a^u) - \overline{G}(X_b^u)$.
23: If $\triangle\overline{G} > \varepsilon$ or $\triangle\overline{G} \leqslant \varepsilon < \tau$, then
24: Replace N by an internal node that splits on X_a^u.
25: For each branch of the split
26: Add a new leaf $\mathcal{L}_j$.
27: Let $ALT(\mathcal{L}_j) = \{\}$.
28: For each $l_k \in \{l_0, l_1\}$ and each $X_i^{un} \in \mathbb{X}^u$
29: Initialize $(\overline{X}_{ik}^{un}, \sigma_{ik}^2, \Sigma_{ik})$ to $(0, 0, 0)$ at $\mathcal{L}_j$.
30: For $l_k \in \{l_0, l_1\}$ and each possible value v_j of each attribute $X_i^{uc} \in \mathbb{X}^u$
31: Initialize $n_{ijk} = 0$ at $\mathcal{L}_j$.

Algorithm 4. Removing an outdated sample from puuCVFDT tree

ForgetSample$(HT, \langle \mathbb{X}^u, y, l\rangle, w_t, ID_w, \zeta)$
// Refer to Algorithm 1 for the details of input parameters.
1: Let N be the root of HT.
2: If $w_t < \zeta$ or$ID(N) < ID_w$, then
3: return.
4: For each $l_k \in \{l_0, l_1\}$ and each $X_i^{un} \in \mathbb{X}^u$
5: Update $(\overline{X}_{ik}^{un}, \sigma_{ik}^2, \Sigma_{ik})$ at N following GA. //Refer to Section 5.2.1
6: For $l_k \in \{l_0, l_1\}$ and each possible value v_j of each attribute $X_i^{uc} \in \mathbb{X}^u$
7: Update n_{ijk} at N following formula (11).
8: For each tree T_{ALT} in *ALT (N)*
9: *ForgetSample*$(T_{ALT}, \langle \mathbb{X}^u, y, l\rangle, w_t, ID_w, \zeta)$.
10: If N is not a leaf, then
11: Split $\langle \mathbb{X}^u, y, l\rangle$ into a set of m fractional samples E. //Refer to Section 4
12: For each sample $s_j \in E$
13: Assign s_j with w_j. //Refer to Section 4
14: Let N_j be the branch child for s_j.
15: *ForetSample*$(N_j, s_j, w_j, ID_w, \zeta)$.

Algorithm 4 lists the pseudo code for removing an outdated sample from the tree. Sufficient statistics are maintained following Gaussian approximation (Steps 4–7) for each node. At an internal node, the original sample is split into m fraction samples, and the subtrees of the node start new removing processes recursively with these fractional samples (Steps 10–15). The alternate trees of each internal node forget an outdated sample analogously (Steps 8 and 9).

6.2. Tree selection

A forest T with nine puuCVFDT trees is trained following Algorithm 1. Periodically, the best estimated tree is chosen as the current output classifier. In function *Estimate* (T, s_t), the following statistical data are incrementally updated as:

1. If s_t is positive, then $|POS| = |POS| + 1$.
2. If s_t is unlabeled, then $|UNL| = |UNL| + 1$.
3. If s_t is positive and it is predicted to be negative by T_k, then $|PToN_k| = |PToN_k| + 1$.
4. If s_t is unlabeled and it is predicted to be positive by T_k, then $|UToP_k| = |UToP_k| + 1$.

In function *GetBestTree (T)*, the following formula is used to evaluate the performance of each $T_k \in T$ [16]:

$$e(T_k) = \frac{|PToN_k|}{|POS|} + \frac{|UToP_k|}{|UNL|},$$

Then, the best tree T_{best} is chosen below [16].

$$T_{best} = argmin(e(T_k)) \qquad 1 \leqslant k \leqslant 9$$

Once the best tree is chosen, the above statistical data is recollected.

6.3. Classifying test samples

A classification model for a test sample $s_t = \langle \mathbb{X}^u, y = ?, l_0\rangle$ is a function that maps attributes $\mathbb{X}^u$ to a probability distribution $\{P(y = y_0), P(y = y_1)\}$. We recursively define $f(s_t, w_t, N)$ to compute such probability distribution. At each internal node N, $f(s_t, w_t, N) = \sum_{j=1}^{m} f(s_j, w_j, N_j)$; while at each leaf node N, $f(s_t, w_t, N) = w_t < p_0, p_1 >$. Here, s_j and N_j represent the jth fractional sample of s_t and the jth child of node N respectively. p_0 and p_1 are obtained following formula (3). Then the class probability distribution of s_t is $f(s_t, 1, \mathcal{R})$. Here, w_t = 1 is the weight for all test samples, $\mathcal{R}$ represents the root of the best tree T_{best}. And s_t is predicted to have class label:

$$\arg\max_{y_k}(P(y = y_0), P(y = y_1)) = \arg\max_{y_k}(f(s_t, 1, \mathcal{R}))$$

7. Experimental study

In this section, we conduct experiments to evaluate the performance of the proposed puuCVFDT algorithm, by comparing with baseline methods:

- **DTU and UDT:** decision tree algorithms for static dataset with uncertainty.
- **POSC4.5:** a PU learning algorithm for static dataset with certain attributes.
- **OcVFDT:** a PU learning algorithm for data stream with certain categorical attributes.
- **UCVFDT:** a supervised learner for data stream with uncertain categorical attributes.
- **CVFDT:** a supervised learner for data stream with certain attributes.

We do not compare with the algorithm proposed in [21], because it tackles label uncertainty, while our puuCVFDT algorithm handles attribute uncertainty. Our evaluation is measured by accuracy and F1 index, which is commonly used by the research community for measuring the performance of positive and unlabeled learning algorithms [6,9].

7.1. Experiment setup

Our algorithms are implemented in Java language based on WEKA[1] and MOA[2] software packages. All experiments are conducted on a computer with Core 4 CPU (2G), 8 GB memory and Linux OS.

Since the unavailability of public uncertain datasets, we conducted our evaluation on synthetic dataset and real-life dataset, by converting them into uncertain ones. This experiment setting is widely used by the research community on classification of uncertain data [23,31,24,13,12].

Synthetic dataset. We use three synthetic datasets in our experiments. (1) Moving hyperplane concept [15,32], it is used to simulate gradual concept drift. We borrow the procedure in [32] to simulate concept drift. In our experiments, 1000 K samples (50% positive and 50% negative) are generated to form the moving hyperplane concept. Each sample consists of d = 10 numerical attributes, two of which are involved in concept changing. (2) SEA concept [30], it is used to simulate abrupt concept drift. In our experiments, 500 K samples (41.9% positive and 58.1% negative) are generated to form SEA concept. (3) Data stream without concept drift, it is generated by using VFML[3] software package, which is used to generate synthetic data stream in [8] by Domingos and Hulten. For simplicity, we name this dataset as VFML concept. In our experiments, 200 K samples (50% positive and 50% negative) are generated to form VFML concept. Each sample consists 100 discriminative attributes with binary attribute value, and 1 category attribute.

Real-life dataset. The Forest CoverType is used in our experiments as real-life dataset. This dataset is about the actual forest cover type for given observation that was determined from US Forest Service Region to Resource Information System.[4] There are 581,012 samples with 7 cover types (class labels) in this dataset. And each sample has 54 attributes, with 10 remotely sensed (numerical) data and 44 cartographic (categorical) data. The majority type, Lodgepole Pine with 283,301 samples, is used as positive class in our experiments. The rest types are used as negative classes.

Simulating uncertainty. We simulate uncertainty (ω) into both numerical and categorical attributes following the approach described in [23,31]. For each sample s_t and for each numerical attribute X_i with original value x_{it}, a random value $\bar{x}_{it}$ selected from $(x_{it} - \omega.x_{it}, x_{it} + \omega.x_{it})$ is used as the mean of $f_{it}(x)$ over $[a_{it}, b_{it}]$. Here, $a_{it} = \bar{x}_{it} - \omega.x_{it}, b_{it} = \bar{x}_{it} + \omega.x_{it}, \omega$ is a user specified parameter, which controls uncertainty of the dataset. For all experiments, uniform distribution is used to generate *pdf*, which implies $f_{it}(x) = 1/(b_{it} - a_{it})$. For categorical attributes, we convert them into probability vectors. For example, when we introduce uncertainty of ω = 10%, categorical attributes will take the original value with 90% probability, and 10% probability to take any of the other values. We use $\omega = 0$ to denote a precise dataset.

Preprocessing. The original version of the above datasets are fully labeled without uncertainty, and they are preprocessed in the following way so as to be feed to the baseline algorithms and the proposed puuCVFDT for performance evaluation.

- **DTU and UDT**. The original datasets with injected uncertainty over all numerical attributes.
- **POSC4.5**. *NumPos%* positive samples are selected randomly from the original dataset as positive training samples, and the rest samples are used as unlabeled.
- **OcVFDT**. As POSC4.5, *NumPos%* positive samples are selected randomly from the original datasets as positive samples, and the rest samples are used as unlabeled.
- **UCVFDT**. The original datasets with injected uncertainty over all categorical attributes.
- **CVFDT**. The original dataset.
- **puuCVFDT**. As POSC4.5, *NumPos%* positive samples are selected randomly from the original dataset as positive samples, and the rest samples are used as unlabeled. Furthermore, these samples are injected with uncertainty over all attributes.

[1] URL: http://www.cs.waikato.ac.nz/ml/weka/.
[2] URL: http://www.cs.waikato.ac.nz/abifet/MOA/.
[3] URL: http://www.cs.washington.edu/dm/vfml/.
[4] URL: http://kdd.ics.uci.edu/databases/covertype/covertype.html.

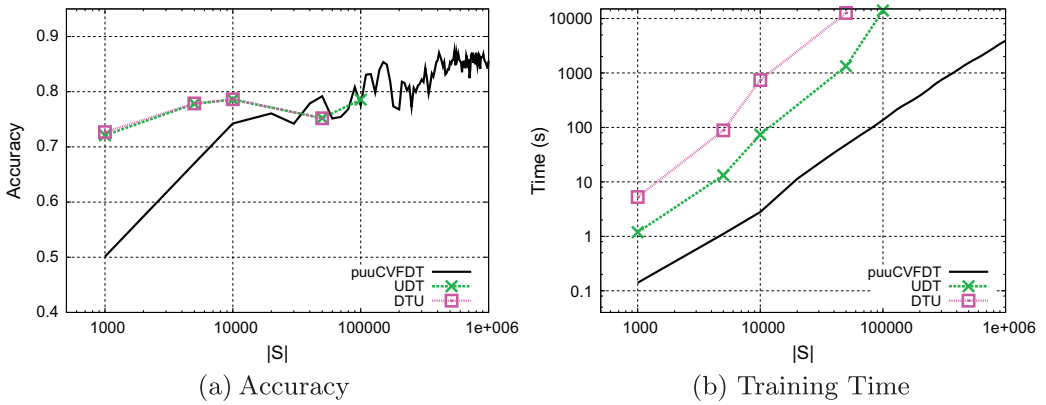

Fig. 1. Ability to handle numerical uncertainty.

To preprocess in the above way makes the puuCVFDT algorithm to face uncertainty and unlabeled samples when it is coping with concept drift in the data stream. And we believe these settings will help to reveal the strong ability of puuCVFDT to learn from positive and unlabeled samples with uncertainty. Note that Hulten et al. did not discuss the way to handle numerical attributes in [15], and we apply the method they released in the source code of their VFML[5] package to our implement of CVFDT to handle numeric attributes. As POSC4.5 is designed for static data, a sliding window version of POSC4.5 is implemented in our experiment by running POSC4.5 on sliding window of 100,000 samples. For puuCVFDT, the slide window also keeps 100,000 samples.

The parameters for CVFDT, OcVFDT, UCVFDT and puuCVFDT are borrowed from literature [15]. For all experiments, we set $\delta = 0.0001$, $f = 20{,}000$, $\tau = 0.05$, $\zeta = 0.005$, $w = 100{,}000$, and $n_{min} = 300$. We also set $Z = 20$, $\mu = 1.96$, $\omega = 30\%$, $M = 20$, $p_{validate} = 0.7$ and $n_{validate} = 1000$ as default value. For each experiment setting, 5 trials of experiments are conducted, and the averaged result is reported.

7.2. Results on synthetic dataset

7.2.1. Ability to handle uncertainty

(A) Uncertain numerical attribute. In this group of experiments, we compare puuCVFDT with DTU and UDT, two state-of-the-art decision tree algorithms for uncertain data, on moving hyperplane concept. We set *NumPos%* = 10% and $\omega = 30\%$. In experiments, at each test point, the upcoming 5000 samples in the stream are used as testing samples. Fig. 1a and b show the accuracy and training time of the experiments respectively. The experiment results of DTU and UDT when $|S| > 100$ K are not reported here, as the experiments could not be finished in two days when $|S| > 100$ K. In Fig. 1a and b, the horizontal axis represents the size of the input data stream. In Fig. 1a, we can observe that DTU and UDT is more accuracy than UCVFDT in early stage (up to 50 K samples). This is because both DTU and UDT make multiple passes over samples to make decisions on each step of tree growing, while puuCVFDT scans each sample once only. However, we can also observe the increasing trend in classification accuracy of puuCVFDT throughout the run, which may suggest that puuCVFDT can cope with uncertainty and reuse the knowledge learned from old samples. It is observed from Fig. 1b that puuCVFDT runs substantially faster than DTU and UDT. The reason is that both DTU and UDT use the multiple passes method to build the tree, while for puuCVFDT, the proposed Gaussian approximation method ensures one-pass over the input data.

We also run with $\omega = 0\%, 10\%, 20\%, 30\%, 40\%$. The classification performance of puuCVFDT with different ω is very close to each other. This demonstrates that puuCVFDT is robust to numerical uncertainty. For clarity, the experiment results of puuCVFDT when $\omega = 0\%, 10\%, 20\%, 40\%$ are omitted here.

(B) Uncertain categorical attribute. Compared with UCVFDT algorithm, we evaluate the ability of puuCVFDT to handle uncertain categorical data on moving hyperplane concept. Numerical attributes are uniformly discretized into 5 bins following the method described in [15]. We run with *NumPos%* = 10%. The uncertainty varies between 0 to 50%. Both algorithms are tested for every 10,000 samples on the upcoming 5000 samples in the data stream, and the averaged experimental results on the whole data stream are reported in Fig. 2. As seen in Fig. 2a and b, when the degree of uncertainty increases, both accuracy and F1 of puuCVFDT decline slowly. But even when data uncertainty reaches 50%, the performance decrement is less than 2%. We can also observe that, under various uncertainty, both accuracy

[5] URL: http://www.cs.washington.edu/dm/vfml/.

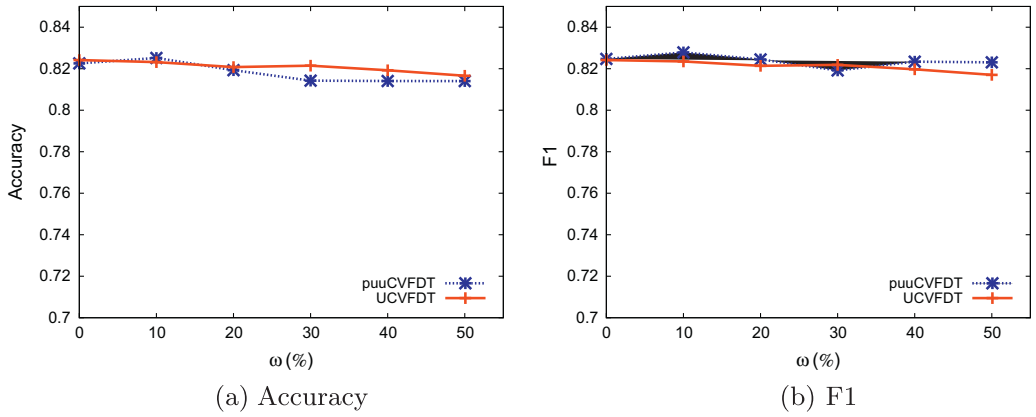

Fig. 2. Ability to handle categorical uncertainty.

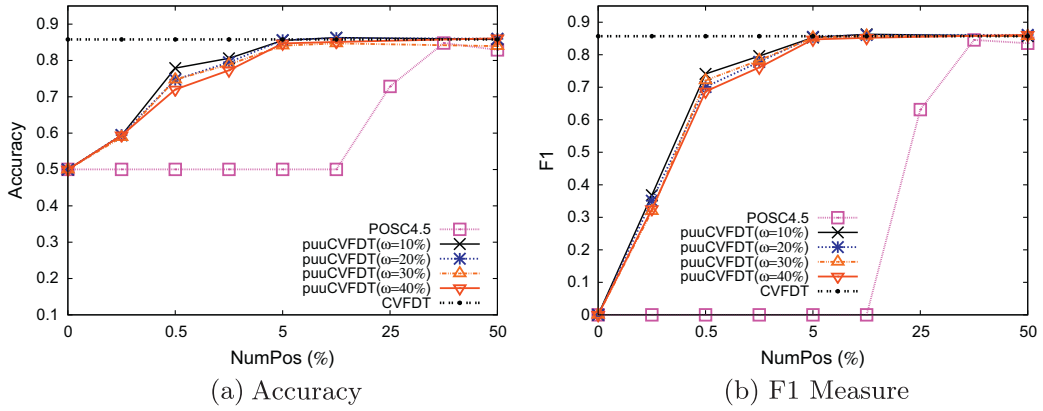

Fig. 3. Ability to learn from positive and unlabeled samples with numerical attributes.

and F1 of puuCVFDT are quite comparable to those of UCVFDT, which is trained on the fully labeled data. This demonstrates that puuCVFDT is also robust to categorical uncertainty.

7.2.2. Ability to learn from positive and unlabeled samples

(A) Numerical data. This group of experiments run on moving hyperplane dataset. We compare the ability of puuCVFDT and POSC4.5 to learn from positive and unlabeled samples with numerical data. Experimental results are obtained following the method described in Section 7.2.1(B). As shown in Fig. 3a and b, the horizontal axis represents the percentage of positive training samples in the data stream, and the vertical axis represents the performance measured in accuracy and F1 respectively. The result of CVFDT is also reported here as the baseline. It is observed from Fig. 3a and b that, in order to learn a robust decision tree, puuCVFDT requires far less positive training samples than POSC4.5. When using *NumPos*% $> 5\%$, the accuracy and F1 of puuCVFDT is as good as that of CVFDT (the reader should note that CVFDT uses fully labeled samples without uncertainty as input). While only when *NumPos*% $> 35\%$, the accuracy and F1 of POSC4.5 is as good as that of CVFDT. We believe this result comes from the fact that the proposed puuCVFDT algorithm has the ability to reuse the knowledge learned from the old samples, which allows it to adapt to current concept in the stream faster than POSC4.5.

It can also be observed that for different degree of uncertainty ($\omega = 10\%, 20\%, 30\%, 40\%$), the accuracy and F1 of puuCVFDT remain quite stable, which demonstrates that puuCVFDT is quite robust against data uncertainty again.

(B) Categorical data. This group of experiments run on VFML concept. We compare puuCVFDT with OcVFDT and POSC4.5 and evaluate the ability to learn from positive and unlabeled samples with categorical data. The generated 200 K samples are randomly divided into two subsets, one of which (100 K) is for training, and the other one (100 K) is for testing. We vary *NumPos%* in training samples and experiment with $\omega = 0$. After observing the whole training data, all models are tested on the whole test data. Experimental results are shown in Fig. 4a and b. It can be also observed that, to learn a robust decision tree, puuCVFDT requires less positive training samples than POSC4.5. The reason is as above.

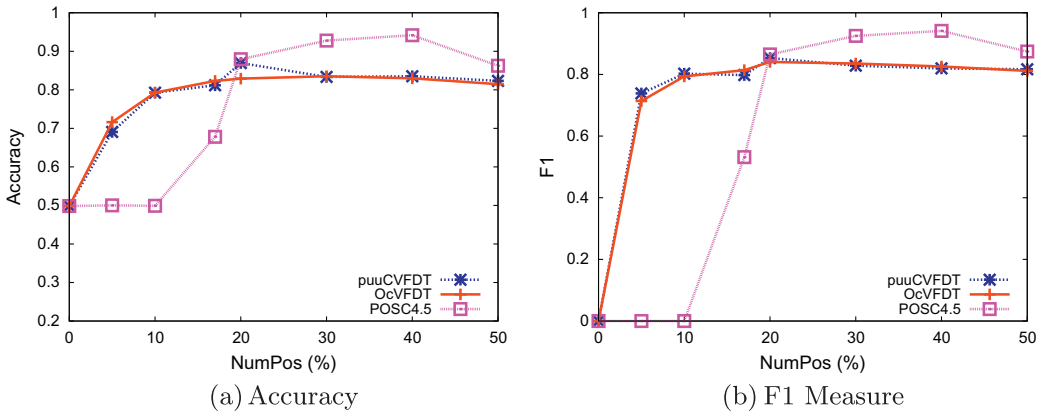

Fig. 4. Ability to learn from positive and unlabeled samples with categorical attributes.

In addition, we can observe that puuCVFDT performs similarly as OcVFDT under various *NumPos%*. This is because both of them can reuse the knowledge learned from the old samples. As there is no concept drift in VFML concept, the mechanism of puuCVFDT to tackle concept drift does not make it outperform OcVFDT.

7.2.3. Ability to cope with concepts drift

(A) Numerical data. In this group of experiments, we evaluate the ability of puuCVFDT to cope with concept drift. Here, we set $\omega = 0$. Gradual drift is simulated on moving hyperplane with *NumPos* = 5%, and the testing is conducted following the methods in Section 7.2.1(B). Abrupt drift is simulated on SEA with *NumPos* = 8%. The learned models are tested every 500 samples on the upcoming 1000 samples in the data stream. Experimental result are shown in Fig. 5a and b. For gradual drift, the probability to observe target concept is kept the same throughout the data stream. So it can be observed from Fig. 5a that puuCVFDT has similar ability as CVFDT to adapt to target concept. For abrupt drift in SEA, the probability to observe target concept also changes suddenly. So it is observed in Fig. 5b that for both puuCVFDT and CVFDT, there is a sudden drop in the accuracy. However, the enumeration of PosLevel from 0.1 to 0.9 makes puuCVFDT converge to current concept faster than CVFDT, which allows the accuracy of puuCVFDT to recover faster.

(B) Categorical data. The similar settings and testing methods as group (A) are used in this group of experiments. We also use the method described in [15] to convert numerical attributes into categorical ones. For moving hyperplane dataset, numerical attributes are uniformly discretized into 5 bins, while for SEA concept, they are uniformly discretized into 10 bins. Let $\omega = 0$. Fig. 6a and b show the performance of CVFDT, OcVFDT and puuCVFDT evaluated on the above two datasets. For CVFDT and puuCVFDT, we can observe the similar result as group (A). This demonstrates that puuCVFDT is also able to cope with concept drift in data stream with categorical data. Fig. 6 also shows that puuCVFDT substantially outperforms OcVFDT. This is because that puuCVFDT has the mechanism to cope with concept drift, while OcVFDT cannot handle concept drift.

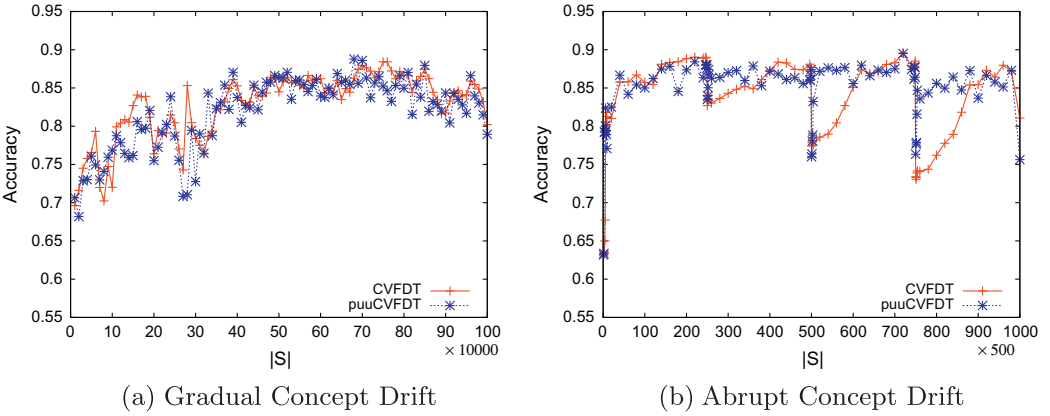

Fig. 5. Ability to cope with concept drift (numerical attributes).

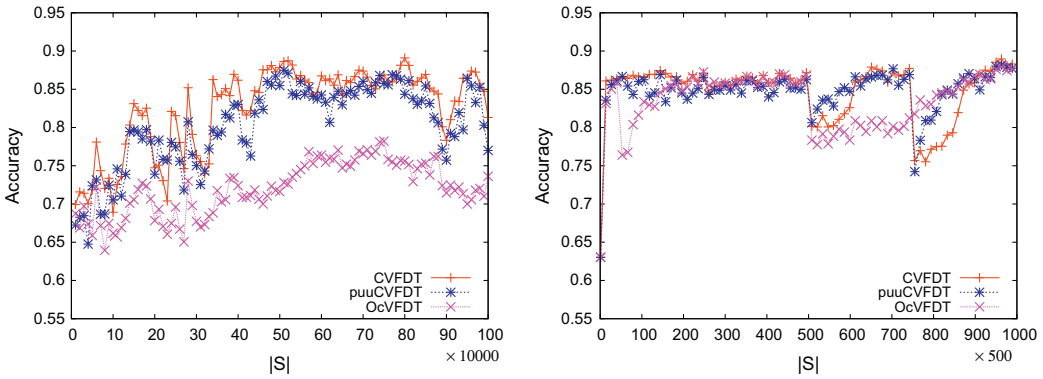

Fig. 6. Ability to cope with concept drift (categorical attributes).

We also set ω = 0%, 10%, 20%, 30%, 40%. For both numerical data and categorical data, the similar results are observed. This demonstrates that puuCVFDT are robust to detect concept drift under various uncertainty. For clarity, we omit the results here.

Compared with CVFDT trained on fully label samples, puuCVFDT shows similar or even better ability to detect concept drift. It can help to save human power greatly, which makes puuCVFDT more applicable to real-life applications.

7.2.4. Experiment with parameters $M, Z, \mu, N_{validate}$ and $P_{validate}$

We conduct experiments on moving hyperplane concept to study how the parameters M, Z, μ, $N_{validate}$ and $P_{validate}$ affect the classification performance of puuCVFDT. We set *NumPos* = 5% and ω = 30%. The testing is conducted following the steps described in Section 7.2.1(B).

Parameters M, Z and μ help to determine the number of samples for approximating $f_{it}(x)$, the number of candidate split point z_i, and a confident interval to help determine the range for candidate z_i respectively. While parameters $N_{validate}$ and $P_{validate}$ help to determine the frequency to select the best tree. The experiment results with various M, Z, μ, $N_{validate}$ and $P_{validate}$ are shown in Fig. 7. In these figures, the horizontal axis represent the value of parameters.

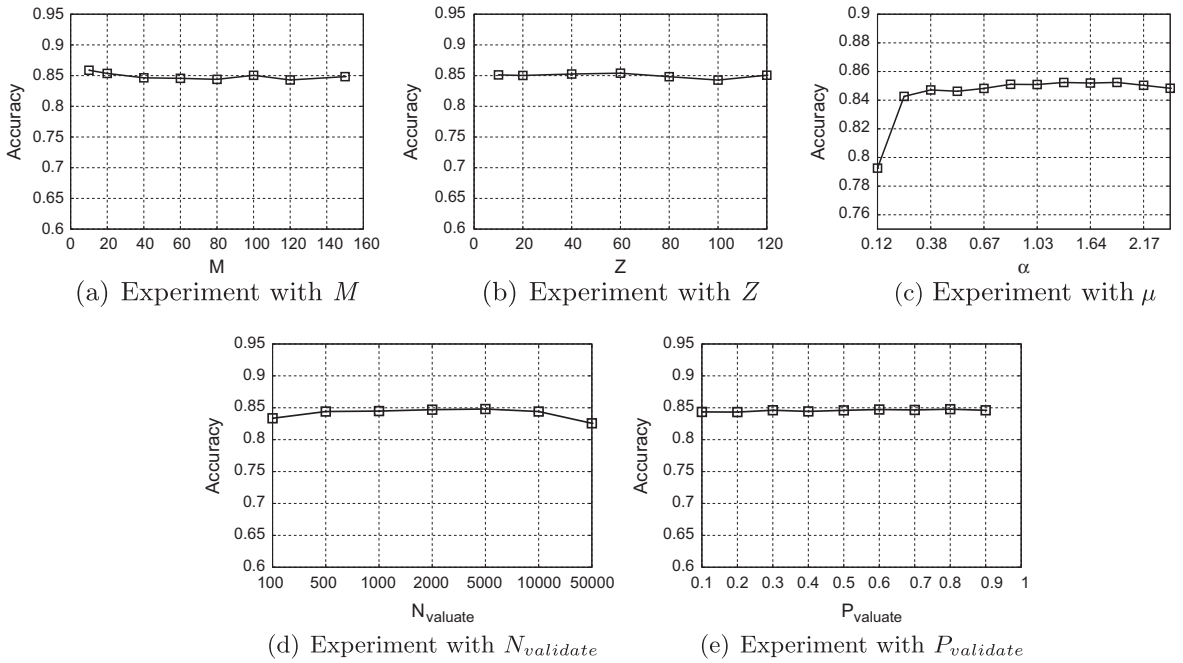

Fig. 7. Experiment with parameter M, Z, μ, $N_{validate}$ and $P_{validate}$.

It is shown in Fig. 7a and b that as M and Z increase, the accuracy of puuCVFDT keeps stable. This shows that our Gaussian approximation method is robust against these two parameters. In Fig. 7c, with increasing of μ, the accuracy increases slightly, but it tends to be stable. This suggests that a larger value of μ with a wider interval for checking split point is not necessary for a better classification performance. The reason is that during training process, there will be further opportunities to refine split decisions on a particular attribute by splitting again further down the tree [1]. In Fig. 7d, we can see if $N_{validate}$ is too small or to large, the accuracy of puuCVFDT will decline. This because a small validate set biases to choose the best tree, while a large one makes the current best tree converge to the current concept too slowly. However, there is a wide range (500–10,000) to choose a proper value for $N_{validate}$. In Fig. 7e, we can see that as $P_{validate}$ increases, the accuracy of puuCVFDT keeps quite stable.

With setting the default value of above parameters, the results of our previous experiments are consistent with the results here.

7.3. Results on the real-life dataset

To evaluate the performance of puuCVFDT on real-life dataset, we conduct a series of tasks to predict Lodgepole Pine type from Forest CoverType data. We make the training samples arrive one by one to form a data stream and for every 5000 samples, we test the accuracy, F1, and training time of learned models on the upcoming 5000 samples throughout the run.

(A) Predict from remotely sensed data. In this group of experiment, we use only remotely sensed data (the 10 numerical attributes, no cartographic data) to predict Lodgepole Pine type. Let *NumPos%* = 25% and $\omega = 0$. Experimental results measured in accuracy, F1, and training time are shown in Fig. 8a, b, and c respectively. It is shown in Fig. 8a and b that puuCVFDT outperforms POSC4.5 on remotely sensed data. In Fig. 8c, it is shown that puuCVFDT runs substantially faster than POSC4.5, which again reveal the high efficiency of the proposed Gaussian approximation method.

We also experiment with parameter ω by setting $\omega = 0\%$, 10%, 20%, 30%, 40%, and the classification accuracy of puuCVFDT is 0.725, 0.716, 0.712, 0.708 and 0.698 respectively. This demonstrates that puuCVFDT is robust to uncertainty again.

(B) Predict from cartographic data. In this group of experiments, we use only cartographic data (the 44 categorical attributes, no remotely sensed data) to predict Lodgepole Pine type. Let *NumPos%* = 25% and $\omega = 0$. Experiment results measured in accuracy, F1, and train time are shown in Fig. 9a, b, and c respectively. It can also be observed from Fig. 9 that puuCVFDT outperforms POSC4.5 and runs substantially faster than POSC4.5 on cartographic data.

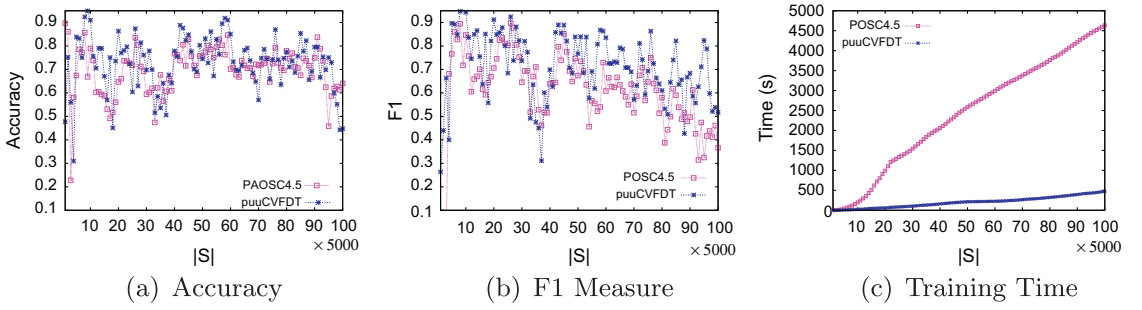

Fig. 8. Results on the remotely sensed data.

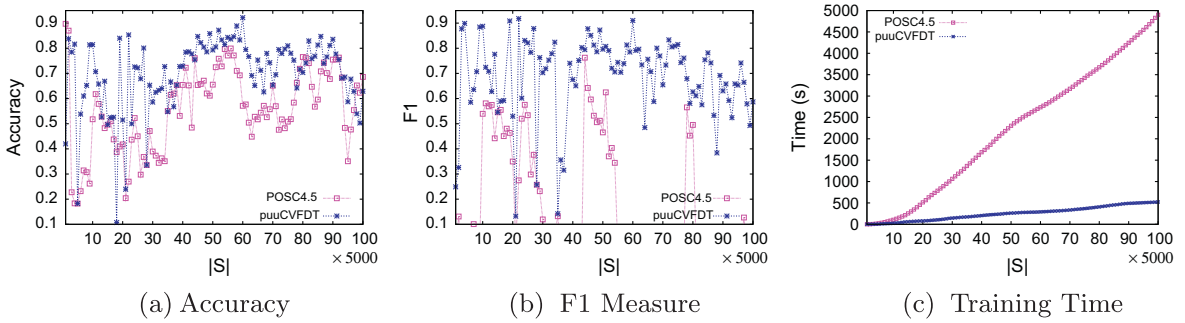

Fig. 9. Results on the cartographic data.

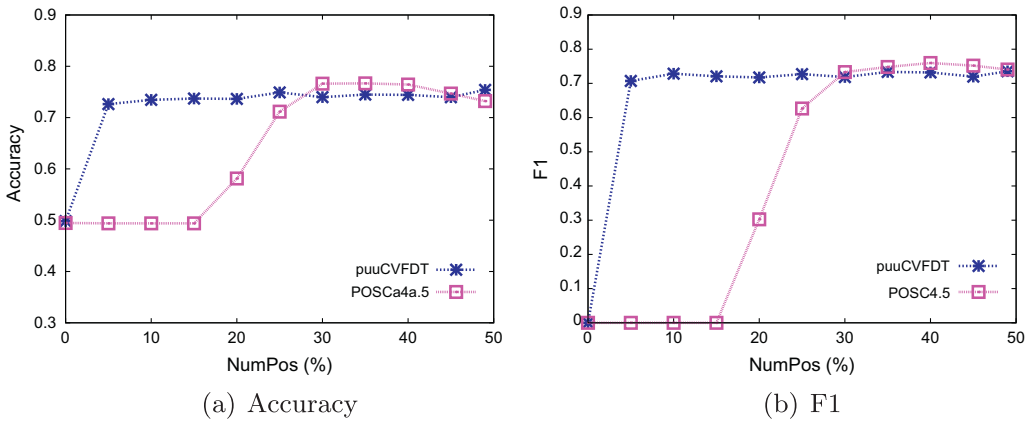

Fig. 10. Results on the Forest CoverType data.

(C) Predict from the whole Forest CoverType data. In this group of experiments, we use the whole Forest CoverType data to predict Lodgepole Pine type. We set $\omega = 0$ and vary the value of *NumPos%*. Averaged experiment results on the whole stream are shown Fig. 10a and b. We can observe that puuCVFDT also requires far less positive training samples than POSC4.5 to build an effective tree on real-life dataset.

Using the same settings as groups (A)–(C), we conduct experiments on OcVFDT. Experimental results are similar as those of puuCVFDT. The reason is as Section 7.2.2(B). There is no concept drift in Forest CoverType, and the ability of puuCVFDT to handle concept drift dose not allow it perform better than OcVFDT. Due to limit space, we omit the results here.

8. Conclusions

In this paper, based on CVFDT, we propose puuCVFDT, a novel very fast decision tree algorithm, to cope with uncertain data streams with positive and unlabeled samples. The main contributions of this paper include: We propose puuIG, uncertain information gain for positive and unlabeled samples. We give methods to summarize the continuous arriving data with imprecise numerical value and/or categorical value into some distributions, and then we use these distributions to calculate puuIG efficiently. We propose probabilistic Hoeffding bound to build the very fast decision trees. And we conduct a series of experiment on both synthetic and real-life dataset. Experimental results show that puuCVFDT has strong abilities to learn from positive and unlabeled samples with uncertainty and cope with concept drift.

The ability to learn from less positive samples with uncertainty make the proposed puuCVFDT algorithm more applicable to real-life applications.

In many real-life applications, the probability to observe the target concept is very low. We plan to study the way to estimate the probability of target concept, so as to mine imbalanced uncertain data streams with only positive and unlabeled samples in the future.

Acknowledgements

This research is supported by the National Natural Science Foundation of China (60873196) and Chinese Universities Scientific Fund (QN2009092). The research is also supported by high-performance computing platform of Northwest A&F University.

References

[1] P. Berbgard, H. Geoffrey, K. Richard, Handling numeric attribute in Hoeffding trees, in: Proceedings of 12th Pacific–Asia Conference on Knowledge Discovery and Data Mining (PAKDD'08), 2008, pp. 296–307.
[2] J. Bi, T. Zhang, Support vector classification with input data uncertainty, in: Advances in Neural Information Processing Systems (SIAM'10), 2010, pp. 161–168.
[3] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, Wadsworth International Group, 1984.
[4] O. Chapelle, B. Schölkopf, A. Zien (Eds.), Semi-Supervised Learning, MIT Press, Cambridge, MA, 2006.
[5] C.A. Charu, S.Y. Philip, A survey of uncertain data algorithms and applications, IEEE Transactions on Knowledge and Data Engineering 21 (2009) 609–623.
[6] F. Denis, R. Gilleron, F. Letouzey, Learning from positive and unlabeled examples, Theoretical Computer Science (2005) 70–83.
[7] X. Ding, X. Lian, L. Chen, H. Jin, Continuous monitoring of skylines over uncertain data streams, Information Sciences 184 (2012) 196–214.
[8] P. Domingos, G. Hulten, Mining high-speed data streams, in: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'00), 2000, pp. 71–80.

[9] C. Elkan, K. Noto, Learning classifiers from only positive and unlabeled data, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'08), 2008, pp. 213–220.
[10] G. Fung, J. Yu, H. Lu, P. Yu, Text classification without negative examples revisit, IEEE Transactions on Knowledge and Data Engineering 18 (2006) 6–20.
[11] B. Gao, T. Liu, W. Wei, T. Wang, H. Li, in: Semi-Supervised Ranking on Very Large Graphs with Rich Metadata (SIGKDD'11), 2011, pp. 1 96–104.
[12] C. Gao, J. Wang, Direct mining of discriminative patterns for classifying uncertain data, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'10), 2010, pp. 861–870.
[13] J. He, Y. Zhang, X. Li, Y. Wang, Naive Bayes classifier for positive unlabeled learning with uncertainty, in: Proceedings of the Tenth SIAM International Conference on Data Mining (SIAM'10), 2010, pp. 361–372.
[14] W. Hoeffding, Probability inequalities for sums of bounded random variables, Journal of the American Statistical Association (1963) 523–528.
[15] G. Hulten, L. Spencer, P. Domingos, Mining time-changing data streams, in: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'01), 2001, pp. 97–106.
[16] C. Li, Y. Zhang, X. Li, One-class very fast decision tree for one-class classification of data streams, in: Proceedings of 3rd International Workshop on Knowledge Discovery from Sensor Data (SensorKDD'09), 2009.
[17] C. Liang, Y. Zhang, Q. Song, Decision tree for dynamic and uncertain data streams, in: Proceedings of 2nd Asian Conference on Machine Learning (ACML2010), 2010, pp. 209–224.
[18] P. Liang, Semi-supervised learning for natural language, Masters thesis, MIT, 2005.
[19] B. Liu, Y. Dai, X. Li, W. Lee, P. Yu, Building text classifiers using positive and unlabeled examples, in: Proceedings of the Third IEEE International Conference on Data Mining (ICDM'03), 2003, pp. 179–186.
[20] J. Liu, X. Li, W. Zhong, Ambiguous decision trees for mining concept-drifting data streams, Pattern Recognition Letters (2009) 1347–1355.
[21] S. Pan, K. Wu, Y. Zhang, X. Li, Classifier ensemble for uncertain data stream classification, in: Proceedings of 14th Pacific–Asia Conference on Knowledge Discovery and Data Mining (PAKDD'10), 2010, pp. 488–495.
[22] S. Pang, T. Ban, Y. Kadobayashi, N. Kasabov, Personalized mode transductive spanning svm classification tree, Information Sciences 181 (2011) 2071–2085.
[23] B. Qin, Y. Xia, F. Li, DTU: a decision tree for uncertain data, in: Proceedings of the 13th Pacific–Asia Conference on Knowledge Discovery and Data Mining (PAKDD'09), 2009, pp. 4–15.
[24] B. Qin, Y. Xia, S. Prabhakar, Y.C. Tu, A rule-based classification algorithm for uncertain data, in: Proceedings of the 25th International Conference on Data, Engineering (ICDE'09), 2009, pp. 1633–1640.
[25] B. Qin, Y. Xia, S. Wang, X. Du, A novel bayesian classification for uncertain data, Knowledge-Based System 24 (2011) 1151–1158.
[26] J. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufman, 1993.
[27] J. Ren, S. Lee, X. Chen, B. Kao, R. Cheng, D. Cheung, Naive Bayes classification of uncertain data, in: Proceedings of the 9th IEEE International Conference on Data Mining (ICDM'09), 2009, pp. 944–949.
[28] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. Smola, R. Williamson, Estimating the support of a high-dimensional distribution, Neural Computation 13 (2001) 1443–1471.
[29] A. Shaker, R. Senge, E. Hllermeier, Evolving fuzzy pattern trees for binary classification on data streams, Information Sciences (2012), http://dx.doi.org/10.1016/j.ins.2012.02.034.
[30] W. Street, 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), 2001, pp. 377–382.
[31] S. Tsang, B. Kao, K.Y. Yip, W.S. Ho, S. Lee, Decision trees for uncertain data, in: Proceedings of the 25th International Conference on Data, Engineering (ICDE'09), 2009, pp. 441–444.
[32] H. Wang, W. Fan, P. Yu, 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), 2003, pp. 226–235.
[33] M. Wang, X. Hua, T. Mei, R. Hong, G. Qi, Y. Song, L. Dai, Semi-supervised kernel density estimation for video annotation, Computer Vision and Image Understanding 113 (2009) 384–396.
[34] D.H.D. West, Updating mean and variance estimates: an improved method, Communications of the ACM 22 (1979) 532–535.
[35] H. Yu, J. Han, K. Chang, PEBL: web page classification without negative examples, IEEE Transactions on Knowledge and Data Engineering 16 (2004) 70–81.
[36] X. Zhu, Semi-Supervised Learning Literature Survey, Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.
[37] F. Zhuang, G. Karypis, X. Ning, Q. He, Z. Shi, Multi-view learning via probabilistic latent semantic analysis, Information Sciences 199 (2012) 20–30.