JMLR: Workshop and Conference Proceedings 13: 209-224
2nd Asian Conference on Machine Learning (ACML2010), Tokyo, Japan, Nov. 8–10, 2010.

Decision Tree for Dynamic and Uncertain Data Streams

Chunquan Liang LCQUAN@NWSUAF.EUD.CN
College of Mechanical and Electric Engineering
Northwest Agriculture and Forest University

Yang Zhang ZHANGYANG@NWSUAF.EDU.CN
College of Information Engineering
Northwest Agriculture and Forest University

Qun Song QSONG@MAIL.NWPU.EDU.CN
Affiliation Department of Automation Department
Northwestern Polytechnical University

Editor: Masashi Sugiyama and Qiang Yang

Abstract

Current research on data stream classification mainly focuses on certain data, in which precise and definite value is usually assumed. However, data with uncertainty is quite natural in real-world application due to various causes, including imprecise measurement, repeated sampling and network errors. In this paper, we focus on uncertain data stream classification. Based on CVFDT and DTU, we propose our UCVFDT (Uncertainty-handling and Concept-adapting Very Fast Decision Tree) algorithm, which not only maintains the ability of CVFDT to cope with concept drift with high speed, but also adds the ability to handle data with uncertain attribute. Experimental study shows that the proposed UCVFDT algorithm is efficient in classifying dynamic data stream with uncertain numerical attribute and it is computationally efficient.

Keywords: Uncertain data streams, Decision Tree, Classification, Concept drift

1. INTRODUCTION

In real-world applications, such as credit fraud detection, network intrusion detection, huge volume of data arrives continuously with high speed. Such applications could be modeled as data stream classification problems. Currently, attribute value is usually assumed to be precise and definite (see, for example, Hulten et al. 2001, Street and Kim 2001, Wang et al. 2003) by the research community of data stream classification analysis. However, data uncertainty arises naturally in many applications due to various reasons, including imprecise measurement, missing values, privacy protection, repeated sampling and network errors, therefore traditional data stream learning algorithms are not applicable to real-life applications.

In the case of credit fraud detection, the raw data of customer such as age, address and vocation may be masked with imprecise values for privacy protection or missing values. For example, the true age 38 may be substituted with a range of [36, 39]; a probability vector $< (doctor, 0.6), (teacher, 0.1), (programmer, 0.3) >$ may be used to replace the true vocation teacher. The same scenario could be observed in network intrusion detection. In this paper, we model these applications as a problem of classifying uncertain data streams.

We identify the following characters of uncertain data stream classification:

1. **Data with uncertainty**: uncertain values have to be handled cautiously, or else the mining results could be unreliable or even wrong.

2. **High speed input data**: the classifier should have the ability to process huge volume of data which arrive at high speed.

3. **Concept-drift detecting**: the classifier should be capable of detecting and responding to changes in the example-generating process.

4. **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 DTU (Qin et al. 2009a) and CVFDT (Hulten et al. 2001), we propose UCVFDT (Uncertainty handling and Concept-adapting Very Fast Decision Tree) algorithm for classifying dynamic and uncertain data streams with high speed. Experimental study on both synthetic and real-life data sets shows that UCVFDT has excellent performance. Even when the uncertainty reaches 20%, the classification performance of UCVFDT is still close to that of CVFDT, which is trained on precise data stream.

This paper is organized as follows. Section 2 reviews the related work. We will introduce our UCVFDT algorithm 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

2.1 Uncertain Data Classification

Numerous uncertain data classification algorithms have been proposed in the literature in recent years. Qin et al. (2009b) proposed a rule-based classification algorithm for uncertain data. Ren et al. (2009) proposed to apply Naive Bayes approach to uncertain data classification problem. Decision tree model is one of the most popular classification models because of its advantages (Tsang et al. 2009, Quinlan 1993). Several decision tree based classifiers for uncertain data are proposed by research community. The well-known C4.5 classification algorithm were extended to the DTU (Qin et al. 2009a) and the UDT (Tsang et al. 2009) for classifying uncertain data. (Qin et al. 2009a) used probability vector and probability density function (pdf) to represent uncertain categorical attribute (Qin et al. 2009b) and uncertain numerical attribute (Cheng et al. 2003) respectively. They constructed a well performance decision tree for uncertain data (DTU). Tsang et al. (2009) used the "complete information" of pdf to construct a uncertain decision tree(UDT) and proposed a series of pruning techniques to improve the efficiency.

Both DTU and UDT algorithm are extensions of C4.5, they can only be used to deal with uncertain static data set, while our UCVFDT algorithm is capable of handling uncertain data stream, in which huge volume of data arrive at high speed.

2.2 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 and CVFDT. The CVFDT improves the VFDT with ability to deal with concept drift. After that, many decision tree based algorithms were proposed for data stream classification (for example, Gama et al. 2005, Gama et al. 2006). For ensemble based approaches, the initial papers use static majority voting (for example, Street and Kim 2001, Wang et al. 2003), while the current trend is to use dynamic classifier ensembles (for example, Zhang and Jin 2006, Zhu et al. 2004). All of algorithms mentioned above can only handle certain data. Pan et al. (2009) proposed two types of ensemble based algorithms, Static Classifier Ensemble (SCE) and Dynamic Classifier Ensemble (DCE) for mining uncertain data streams. To the best of our knowledge, this is the only work devoted to classification of uncertain data streams. However, in (Pan et al. 2009), class value of a sample is assumed to be uncertain, while attributes is assumed to have precise value. Our UCVFDT handles uncertain attributes, while class value is assumed to be precise.

3. UCVFDT

3.1 Problem Definition

In this paper, we only consider data streams with categorical attribute values. For data streams with numerical attribute values, numerical attributes could be discretized to categorical attributes using the approach described in (Fayyad and Irani. 1992).

Uncertain attribute model. In this paper, we adopt the model described in (Qin et al. 2009b) to represent uncertain categorical attribute (UCA), which has definition as follows:

Definition 1. (Qin et al. 2009b) An uncertain categorical attribute is denoted by $X_i^{u_c}$, $X_{it}^{u_c}$ is used to denote the t-th sample of $X_i^{u_c}$. Given a categorical domain $Dom(X_i^{u_c}) = \{v_1, \ldots, v_m\}$, $X_i^{u_c}$ is characterized by probability distribution over Dom. It can be represented by the probability vector $P =< p_1, \ldots, p_m >$ such that $P(X_i^{u_c} = v_j) = p_j$ and $\sum_{j=1}^{m} p_j = 1$.

For a certain categorical attribute X_i, we assume sample s_t has attribute value $X_i = v_j$. It can be modeled as a special case of uncertain categorical attribute with $P(X_i = v_j) = 1$, for all $v \neq v_j$, we have $P(X_i = v) = 0$. For simplicity, all attributes will be modeled as uncertain categorical attributes in rest of this paper.

Uncertain data stream classification. Given a stream data $S = \{s_1, s_1, \ldots, s_t, \ldots\}$, where s_t is a sample in S, $s_t = \langle X^{u_c}, y \rangle$. Here $X^{u_c} = \{X_1^{u_c}, X_2^{u_c}, \ldots, X_d^{u_c}\}$ is an attribute set with d uncertain attributes. $y \in C$ denotes the class label of s_t, where C is a set of class labels on stream data. We build decision tree model $y = f(X^{u_c})$ from S to classify the unknown samples.

3.2 Fractional Sample

Note that uncertain tree growing is a recursive process of partitioning the training samples into fractional samples (Qin et al. 2009a, Tsang et al. 2009). In this paper, we also adopt the technique of fractional sample to handle uncertainty. Based on (Quinlan 1993, Tsang et al. 2009), fractional sample is define as follows.

Definition 2. Assume the split attribute at node N is denoted by $X_i^{u_c}$, sample s_t is split into new samples $\{s_{t1}, s_{t2}, \ldots, s_{tm}\}$ by $X_i^{u_c}$, $m = |Dom(X_i^{u_c})|$, where s_{tj} is copy of s_t except for attribute $X_i^{u_c}$, which is set $P(X_{ij}^{u_c} = v_j) = 1$, for all $v \neq v_j$, $P(X_{ij}^{u_c} = v) = 0$. Thus s_{tj} is a fractional sample of s_t on attribute $X_i^{u_c}$.

We associate s_t with a weight w_t, which can be interpreted as the exist probability at node N. The weight of s_{tj} is assigned with

$$w_{tj} = w_t * P(X_{ij}^{u_c} = v_j) \tag{1}$$

which can be interpreted as the probability that s_{tj} falls into the j-th child of node N. All samples at root are set $w_t = 1$. It is considered that s_{tj} exist with low probability if w_{tj} is small. Hence, for a given threshold ζ, if $w_{tj} < \zeta$ is observed, s_{tj} will not be split into fractional samples.

3.3 Uncertain Information Gain

Information Gain (IG) is widely used in decision tree algorithms to decide which attribute being selected as the next splitting attribute. Qin et al. (2009a) give the definition of uncertain information gain (UIG) for building decision tree for uncertain data (DTU). Base on DTU, for a data set S denoted by node N and an attribute set $X^{u_c} = \{X_1^{u_c}, X_2^{u_c}, \ldots, X_d^{u_c}\}$, formula (2) is used to compute UIG.

Definition 3. Uncertain Information Gain (Qin et al. 2009a):

$$UIG(S, X_i^{u_c}) = Entropy(S) - \sum_{j=1}^{m} \frac{PC(S_{ij})}{PC(S)} \times Entropy(S_{ij}) \tag{2}$$

Here, $m = |Dom(X_i^{u_c})|$. $Entropy(S)$ denotes expected information entropy, which can be defined as follows (Qin et al. 2009a):

$$Entropy(S) = -\sum_{k=1}^{|C|} P_S(y_k) \log P_S(y_k) \tag{3}$$

Here, $P_S(y_k)$ denotes the ratio of the sum of probabilities of each sample in y_k to the sum of probabilities of each sample. $P_S(y_k)$ can be computed by (Qin et al. 2009a):

$$P_S(y_k) = \frac{PC(S, y_k)}{PC(S)} \tag{4}$$

Here, PC denotes the sum of probabilities of each sample in S, which is named probabilistic cardinality(Qin et al. 2009a). Based on (Qin et al. 2009a), probabilistic cardinality is defined as follows:

$$PC(S) = \sum_{t=1}^{|S|} w_t \tag{5}$$

S_{ij} denotes the set of samples in S that $X_i^{u_c} = v_j$, $PC(S, y_k)$ denotes the sum of probabilities of each sample in y_k. Here we have

$$PC(S_{ij}) = \sum_{t=1}^{|S|} w_t \times P(X_{it}^{u_c} = v_j) \tag{6}$$

$$PC(S, y_k) = \sum_{t=1}^{|S|} w_t \times P(C(s_t) = y_k) \tag{7}$$

At each node of decision tree, for each possible value $v_j \in Dom(X_i^{u_c})$ of each attribute $X_i^{u_c}$, for each class y_k, we associate it with a sufficient statistic n_{ijk}. In order to collect statistical data for formula (5), (6) and (7), we only need to update n_{ijk} for each new coming sample s^{new} as follows:

$$n_{ijk} = n_{ijk} + PC(\{s^{new}\}_{ij}, y_k) \tag{8}$$

Similarly, for removing a sample s^{old} from node N, we update n_{ijk} as follows:

$$n_{ijk} = n_{ijk} - PC(\{s^{old}\}_{ij}, y_k) \tag{9}$$

For the sake of efficiency and limited memory available, we do not save all the samples observed from the stream for multiple scans

3.4 CVFDT

Hulten et al. (2001) proposed to choose the best split attribute when building decision tree for data streams based on Hoeffding bound (Hoeffding 1963). After n independent observation of a real-valued random variable r with range R, the Hoeffding bound ensures that, with confidence $1 - \delta$, the true mean of r is at least $\overline{r} - \varepsilon$, where $\overline{r}$ is the observed mean of the samples and $\varepsilon = \sqrt{R^2 ln(1/\delta)/2n}$. Let $G(X_i)$ be the evaluation function used to choose test attributes. Assume G is to be maximized, and let X_a be the attribute with highest observed $\overline{G}$ after seeing n examples and X_b be the second-best attribute. Let $\Delta\overline{G} = \overline{G}(X_a) - \overline{G}(X_b)$. Given a desired δ, the Hoeffding bound guarantees that X_a is the correct choice with probability $1-\delta$ if n examples have been seen at this node and $\Delta\overline{G} > \varepsilon$. Then the node can be split using the current best attribute, and succeeding examples will be passed to the new leaves.

According to 3.2, a sample s_t exists at node N with probability w_t in uncertain decision tree. For uncertain data, the count for n observation at node N is represented by expected count, which is defined as follows:

$$n = \sum_{t=1}^{|S|} w_t = PC(S) \tag{10}$$

Here S denotes fractional samples observed at node N, so we have

$$\varepsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2PC(S)}} \tag{11}$$

We name Hoeffding bound following formula (11) as probabilistic Hoeffding bound.

3.5 Building UCVFDT

Here, based on probabilistic Hoeffding bound, we use the uncertain information gain as the split evaluation function to choose the best splitting attribute. We extend CVFDT to our UCVFDT (Uncertainty-handling and Concept-adapting Very Fast Decision Tree) algorithm. Like CVFDT with the capability of detecting concept drift, UCVFDT works by keeping its model consistent with a sliding window of uncertain samples. The pseudo code is illustrated in Algorithm 1. Function $G(.)$ denotes uncertain information gain. ζ is a threshold for fractional sample, $(X^{u_c})_l$ denotes uncertain attribute set at node l, n_{ijk} denote the sufficient statistics for computing uncertain information gain. Please refer to (Hulten et al. 2001) for details of other parameter.

In algorithm 1, the process for handling a sample could be summarized into four parts. Firstly, from step 11 to step 15, the new coming sample is associated with an ID and a weight, then it is saved to the sliding window. Secondly, from step 16 to step 20, an outdated sample is remove from the tree, it is also deleted from the slide window (Algorithm 3). Thirdly, step 21 is for tree growing (Algorithm 2). Lastly, step 22 and step 23 is for checking split validity of an internal node periodically, please refer to (Hulten et al. 2001) for more details.

Algorithm 2 lists the pseudo code for growing uncertain decision tree. Sufficient statistics are collected from fractional samples from step 5 to step 9. From step 14 to step 19, a sample is split into a set of fractional samples. From step 21 to 39, the uncertain information gain is computed using sufficient statistics at leaf nodes. Based on probabilistic Hoeffding bound, split attribute is chose and the leaf node is split into an internal node.

Algorithm 3 lists the pseudo code for removing an outdate sample from uncertain decision tree. The original sample is split into a set of fractional samples, and sufficient statistics is updated by subtracting probabilistic cardinality of all these fractional samples.

3.6 Classifying test sample

Given an uncertain test sample $s_t =< X^{u_c}, ? >$, a classification model for uncertain data is a function that maps attributes X^{u_c} to a probability distribution vector $< P(y_1, y_2, \ldots, y_{|C|}) >$ (Qin et al. 2009a, Tsang et al. 2009). For computing such probability distribution, we recursively defined $f_N(s_t, w_t) = \sum_{j=1}^{m} f_{(N,j)}(s_{tj}, w_{tj})$ at node N. Here, (N, j) represents the j-th sub tree of node N. If node N is leaf node l, we define $f_l(s_{tj}, w_{tj}) = w_{tj}$. $< PC_S(y_1), PC_S(y_2), \ldots, PC_S(y_{|C|}) >$. Here, S represents samples observed by node l. Finally, we compute class probability distribution for s_t using $f_r(s_t, w_t)$, where r denotes the root of UCVFDT, and set $w_t = 1$ for all test sample. The test sample will be predicted to be of class y_k which has the largest $P(y_k)$, $k = 1, 2, ...|C|$.

Algorithm 4 lists pseudo code for classifying an uncertain test sample s_t. In algorithm 4, $dist =< PC_S(y_1), PC_S(y_2), \ldots, PC_S(y_{|C|}) >$, $l.dist = f_l(s_{tj}, w_{tj})$ is probability distribution over classes at node l.

3.7 Time Complexity and Space Complexity

Like CVFDT, The memory requirement by UCVFDT are dominated by the sufficient statistics and are independent of the total number of samples seen. It is able to keep the model in memory proportional to $O(hdmc)$. Here, h denotes the number of nodes in uncertain deci-

Algorithm 1 the learning algorithm for UCVFDT

```
Inputs:
    $S$      a stream of samples,
    $X^{u_c}$  an uncertain categorical attribute vector,
    $G(.)$   uncertain information gain for split evaluation,
    $\delta$      one minus the desired probability of choosing the correct attribute at any given
  node,
    $\tau$      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.
    $\zeta$      a user-supplied weight threshold,
Outputs:
    $HT$    a decision tree for uncertain samples.
 1: Let $HT$ be a tree with a single leaf $l_1$ , the root.
 2: Let $ALT(l_1)$ be an initially empty set of alternate trees for $l_1$.
 3: Let $\overline{G}(X_\phi)$ be the $\overline{G}$ obtained by predicting the most frequent class for $S$ following
    formula (7).
 4: Let $(X^{u_c})_1 = X^{u_c} \bigcup \{X_\phi\}$.
 5: Let $W$ be be the window of sample, initially empty.
 6: for each class $y_k$ do
 7:   for each possible value $v_j$ of each attribute $X_i^{u_c} \in X^{u_c}$ do
 8:     Let $n_{ijk} = 0$.
 9:   end for
10: end for
11: for each sample $(X^{u_c}, y)$ in $S$ do
12:   Sort$(X^{u_c}, y)$ into a set of leaves $L$ using $HT$ of any node $(X^{u_c}, y)$ passes through.
13:   Let $ID$ be the maximum id of the leaves in $L$.
14:   Weight $(X^{u_c}, y)$ with value 1.
15:   Add $((X^{u_c}, y), ID)$ to the beginning of $W$.
16:   if $|W|$ >w then
17:     Let $((X_w^{u_c}, y_w), ID_w)$ be the last element of $W$.
18:     ForgetSamples($HT$,$(X_w^{u_c}, y_w), ID_w$,$\zeta$).      //refer to Algorithm 3
19:     Let W=W with $((X_w^{u_c}, y_w), ID_w)$ removed.
20:   end if
21:   UCVFDTGrow($HT$,G,$(X^{u_c}, y)$,$\delta$,$n_{min}$,$\tau$,$\zeta$).      //refer to Algorithm 2
22:   if If there have been $f$ samples since the last checking of alternate trees then
23:     CheckSplitValidity($HT$,$\delta$).
24:   end if
25: end for
26: Return $HT$.
```

Algorithm 2 UCVFDTGrow(HT,G,(X^{u_c}, y),δ,n_{min},τ,ζ)

```
// Refer to Algorithm 1 for the details of input parameters;
1: if the weight of $(X^{u_c}, y) < \zeta$ then
2:    Return.
3: end if
4: Let $l$ be the root of $HT$.
5: for each class $y_k$ do
6:    for each possible value $v_j$ of each attribute $X_i^{u_c} \in X^{u_c}$ do
7:       Increment $n_{ijk}(l)$ following formula (8).
8:    end for
9: end for
10: for each tree $T_{ALT}$ in $ALT(l)$ do
11:    UCVFDTGrow($T_{ALT}$,$G$,$(X^{u_c}, y)$,$\delta$,$n_{min}$,$\tau$,$\zeta$).
12: end for
13: Label $l$ with the majority class among the samples seen so far at $l$ following formula
    (7).
14: if $l$ is not a leaf then then
15:    Split $(X^{u_c}, y)$ into a set of fractional samples $E$.
16:    for each sample $s_{tj}$ in $E$ do
17:       Reweight $s_{tj}$ following formula (1).
18:       Let $l_j$ be the branch child for $s_{tj}$.
19:       UCVFDTGrow($l_j$,$G$,$s_{tj}$,$\delta$,$n_{min}$,$\tau$,$\zeta$).
20:    end for
21: else
22:    Let $n_1$ and $n_2$ be be the expect count of samples last seen and current seen at $l$
       computed by formula (10).
23:    if If the samples seen so far at $l$ are not all of the same class and $n_1 - n_2 > n_{min}$
       then
24:       Compute $G(X)$ for each attribute $X_i^{u_c} \in X^{u_c} - \{X_\phi\}$ using $n_{ijk}(l)$ and formula
          (2).
25:       Let $X_a^{u_c}$,$X_b^{u_c}$ be the attribute with highest and second-highest $\overline{G}$.
26:       Compute $\varepsilon$ using formula (11) and $\delta$.
27:       Let $\triangle\overline{G} = \overline{G}(X_a^{u_c}) - \overline{G}(X_b^{u_c})$.
28:       if $\triangle\overline{G} > \varepsilon$ or $\triangle\overline{G} <= \varepsilon < \tau$ then
29:          Replace $l$ by an internal node that splits on $X_a^{u_c}$.
30:          for each branch of the split do
31:             Add a new leaf $l_j$, and $(X^{u_c})_j = X^{u_c} - \{X_a^{u_c}\}$.
32:             Let $ALT(l_j) = \{\}$.
33:             for each class $y_k$ and each $v_j$ possible value of each attribute $X_i^{u_c} \in (X^{u_c})_j$
                do
34:                Let $n_{ijk}(l_j) = 0$.
35:             end for
36:          end for
37:       end if
38:    end if
39: end if
```

Algorithm 3 ForgetSamples(HT,$(X_w^{u_c}, y_w), ID_w$,ζ)

```
 // Refer to Algorithm 1 for the details of input parameters;
 1: Let $l$ be the root of $HT$.
 2: if $(X_w^{u_c}, y_w)$'s weight $< \zeta$ or $l.ID > ID_w$ then
 3:    Return.
 4: end if
 5: for each class $y_k$ do
 6:    for each possible value $v_j$ of each attribute $X_i^{u_c} \in X_w^{u_c}$ do
 7:       Decrement $n_{ijk}(l)$ following formula (9).
 8:    end for
 9: end for
10: for each tree $T_{alt}$ in $ALT(l)$ do
11:    ForgetSamples($T_{alt}$,$(X_w^{u_c}, y_w), ID_w$,$\zeta$).
12: end for
13: if $l$ is not a leaf then
14:    Split $(X_w^{u_c}, y_w)$ into a set of fractional sample $E$.
15:    for each sample $s_{wj}$ in $E$ do
16:       Reweight $s_{wj}$ following formula (1).
17:       Let $l_j$ be the branch child for $s_{wj}$.
18:       ForgetSamples($T_{alt}$,$s_{wj}$,$ID_w$,$\zeta$).
19:    end for
20: end if
```

Algorithm 4 ClassifySample(HT,s_t)

```
Inputs:
        HT     a decision tree for uncertain data,
        $s_t$      a test sample,
Outputs:
        dist   probabilistic distribution over C.
 1: Let $l$ be the root of HT.
 2: if is $l$ leaf node then then
 3:    $dist = dist + l.dist$
 4: else
 5:    Split $s_t$ into a set of fractional sample $E$.
 6:    for each $s_{tj}$ in E do
 7:       Reweight following formula (1).
 8:       Let $l_j$ be the branch child for $s_{tj}$.
 9:       ClassifySample($l_j$,$s_{tj}$).
10:    end for
11: end if
```

sion tree and alternate trees, d denotes the number of attributes, $m = max_{i=1}^{d}|Dom(X_i^{u_c})|$ and c denotes the number of class label. In order to keep model up-to-date, the time complexity of UCVFDT is $O(l_u dmc)$, where l_u denotes the number of nodes in uncertain decision tree and alternate trees, while CVFDT's is $O(l_c dmc)$, where l_c denotes the length of the longest path an sample will have to take through decision tree times the number of alternate trees. Apparently, $l_u >= l_c$. It is concluded that UCVFDT is more time-consuming than CVFDT for handling uncertainty. We will compare running time of UCVFDT to CVFDT's in our experiment.

4. EXPERIMENT STUDY

In order to measure the classification performance of UCVFDT, we perform experiments on both synthetic data set (Moving hyperplane (Hulten et al. 2001), SEA (Street and Kim 2001)) and real-life data set (CoverType[1]).

Since no benchmark uncertain data set can be found in the literature, we simulate uncertainty (u%) by converting categorical attributes into probability vectors using the approach described in (Qin et al. 2009b). For example, when we introduce uncertainty of 10%, attributes will take the original value with 90% probability, and 10% probability to take any of the other values. A data set with u% in categorical data is denoted by Uu. We use Ud for the number of attributes which are made uncertain.

Our algorithms are implemented in Java language based on WEKA[2] and MOA[3] software packages, and the experiments are conducted on a PC with Core 2 CPU, 2G memory and Windows XP OS. For all of experiments, the parameters of UCVFDT were set as follows: $\delta = 0.0001$, $f = 20000$, $\tau = 0.05$, $\zeta = 0.0005$, w $= 100000$, $n_{min} = 300$. The parameters of CVFDT are the same as those of (Hulten et al. 2001).

4.1 Experimental Results in Moving Hyperplane

Moving hyperplane have been widely used to simulate time-changing concepts (for example, Hulten et al. 2001, Wang et al. 2003). A moving hyperplane in d-dimensional space is denoted by: $\sum_{i=1}^{d} a_i x_i = a_0$. We followed the same procedure in (Wang et al. 2003) to simulate drift. We used K for the total number of dimensions whose weights involved in concept changing, Parameter $q \in R$ and $s_i \in \{1, -1\}$ specify the magnitude and direction of the change (every M samples) for weights $a_1, a_w, \ldots, a_k$. For more detailed descriptions of moving hyperplane, please refer to (Wang et al. 2003).

In our experiment, we set $M = 1000$, numerical attributes are uniformly discretized into 5 bins using the method described in (Fayyad and Irani. 1992). Experiment results are obtained by testing the accuracy and training time of models every 10,000 samples throughout the run using the subsequent 5,000 sample as the test data.

1. URL: http://www.kdd.ics.uci.edu.
2. URL: http://www.cs.waikato.ac.nz/ml/weka/.
3. URL: http://www.cs.waikato.ac.nz/ abifet/MOA/.

4.1.1 UCVFDT vs. DTU

We examine how well UCVFDT would compare to a system using traditional drift-tracking methods. We thus compare UCVFDT, DTU-Window. We simulate DTU-Window by running DTU on W for every 100,000 examples instead of for every example. We experiment with $d = 30$, $K = 5$, $Uu = U0, U5, U10, U15, U20$. $Ud = K = 5$. Stream size is set $|S| = 1,000,000$ with noise p=5%. Fig.1, Fig.2 and table 1 show experiment results.

Comparison of Classification Accuracy: Fig.1 shows the accuracy of UCVFDT, DTU-Window and CVFDT. Here CVFDT is a special case of UCVFDT that runs on data set with $U0$. In Fig.1, and vertical axis represents classification accuracy. Fig.1(A), Fig.1(B) and Fig.1(C) gives the experiment result for $U10$, $U15$ and $U20$ respectively. It can be observed that DTU-Window is better than UCVFDT when q is small, but with increasing of q, which implies concept drift dramatically, UCVFDT outperform DTU-Window. CVFDT performs the best, because CVFDT is trained on precise data, from which more information is acquired. Fig.2 shows a detailed view of one of the runs from Fig.1, the one for $U20$. In Fig.2, horizontal axis represents parameter number samples. It can be observed from Fig.2 that DTU-Window is close to UCVFDT. But DTU-Window is at a cost of 72.5 times of UCVFDT on training time spending. More details about training time will be analyzed in the following paragraph.

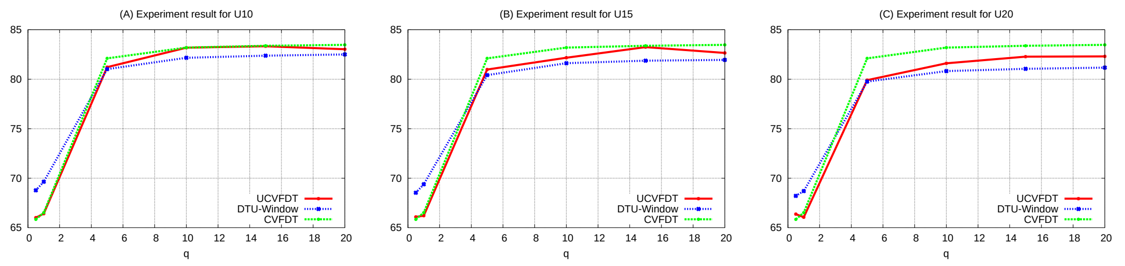

Figure 1: Comparison of UCVFDT ,DTU-Window and CVFDT by Accuracy.

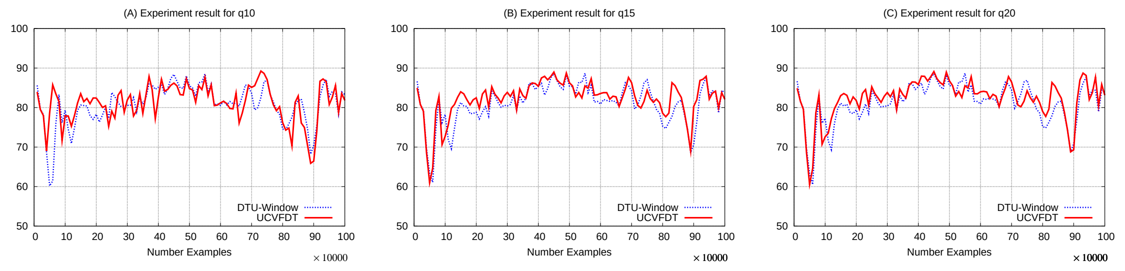

Figure 2: Accuracy as a function of the number of samples seen.

Running Time and Tree Size: We compare training time and tree size of UCVFDT to DTU-Window's. Table 1 shows a detailed view for Fig.1(B), Fig.1(C), Fig.1(D). The differences in model size and run times are very large. The average model size of UCVFDT is about 229, and DTU-Window's is 3105; The average training time of UCVFDT is 1.72s, DTU-Window's is 124.81s. DTU-Window is more time-consuming than UCVFDT, because

DTU-Window have to train on examples in W from scratch every time, while UCVFDT learn from a sample in an incremental way. Overall speaking, UCVFDT achieve better accuracy with far fewer nodes and training time than DTU-Window.

Table 1: Tree Size and Running Time.

Uu	q	UCVFDT			DTU-Window		
		nodes	leaves	training time (s)	nodes	leaves	training time(s)
U10	0.5	148.4	118.9	0.67	8988.7	7191.2	6.04
	1	186.5	150	1.6	3752.1	3001.8	42.8
	5	282.2	226.6	2.18	1167.5	934.2	146.83
	10	260.4	208.8	1.74	1136.2	909.2	145.49
	15	284	228	2.38	1141.8	913.6	160.06
	20	276.5	221.9	2.37	1141	913	165.75
U15	0.5	148.4	118.9	0.67	9487.1	7589.8	5.65
	1	169.1	135.9	1.47	3451.8	2769.4	42.2
	5	278.6	223.5	2.14	1271.6	1017.4	167.53
	10	306.3	246.5	1.7	1325.9	1060.9	167.53
	15	275.9	221.3	2.25	1339.5	1071.7	180.19
	20	259.6	208.3	2.16	1354.5	1083.7	185.49
U20	0.5	148.3	118.8	0.69	10088.4	8070.9	5.19
	1	155.9	125.2	1.28	3893.2	3114.8	38.46
	5	233.1	186.8	2.01	1509.3	1207.6	197.75
	10	211.4	170.2	1.28	1589.3	1271.6	193.95
	15	242.2	194.3	2.21	1618.2	1294.7	199.59
	20	254.4	204.2	2.19	1625.5	1300.5	196.15

4.1.2 UCVFDT vs. CVFDT

In this group of experiments, we compare the classification performance of UCVFDT to CVFDT. For stream data we set $d = 10$, $K = 2, 4, 6, 8$, $Ud = K = 2, 4, 6, 8$, $|S| = 1,000,000$ with noise p=5%. CVFDT run with data stream without uncertainty. Experiment results show that UCVFDT and CVFDT induce trees with similar number of nodes, the average tree size of UCVFDT is about 285 while CVFDT's is about 340. Training time needed by UCVFDT averages out to 1.8s, while CVFDT averages out to 0.4s. It can be concluded that UCVFDT take more time to handle uncertain information. For lacking of space, we only report accuracy in detail. Table 2 gives the accuracy of CVFDT and UCVFDT with $Uu = U5, U10, U15, U20, U25$ and $q = 1, 5, 10$; Overall speaking, the accuracy of UCVFDT remains relatively stable. Even when the extent of uncertainty reaches 25%, the accuracy is still comparable to CVFDT over certain or precise data. It shows that UCVFDT is robust against data uncertainty.

Table 2: Comparison of UCVFDT and CVFDT by Accuracy.

K(Ud)	q	CVFDT	UCVFDT				
			U5	U10	U15	U20	U25
2	1	79.95	79.96	79.75	79.83	79.89	79.73
	5	84.82	85.1	85.1	85.06	84.01	83.79
	10	85.48	85.48	85.3	85.27	85.23	84.86
4	1	71.75	71.45	71.42	71.04	71.15	71.35
	5	81.2	80.86	81.32	80.59	80.39	80.7
	10	82.48	81.89	81.65	81.86	81.62	81.26
6	1	75.18	74.75	74.06	73.89	73.53	73.13
	5	78.44	76.95	78.53	78.2	78.41	77
	10	79.44	79.24	78.79	78.34	78.19	77.72
8	1	78.57	78.39	77.75	77.38	77.18	76.71
	5	79.91	79.2	79.03	78.83	78.15	78.72
	10	79.48	79.06	78.76	78.02	77.46	77.45

4.2 SEA

We examine UCVFDT's abrupt concept drift mechanism by comparing with CVFDT using another synthetic data SEA. Here, we do not compare UCVFDT with DTU, because DTU can not handle data stream. SEA is described in (Street and Kim 2001) and it is used for generate abrupt concept drift data set. It is generated using three attributes, where only the two first attributes are relevant. The points of the data set are divided into 4 blocks with different concept. In each block, the classification is done using $f_1 + f_2$, where f_1 and f_2 represent the first two attributes and θ is a threshold value. In this group experiment we set $\theta = 9, 8, 7$ and 9.5 for the data blocks. Numerical attributes were discretized into 10 bins, and were simulated uncertain information with $Uu = 10\%, 20\%$ and 30%. We set stream data $|S| = 500,000$ with p=10% noise. After every 500 training samples, we use the subsequent 1000 sample as test data to test the model.

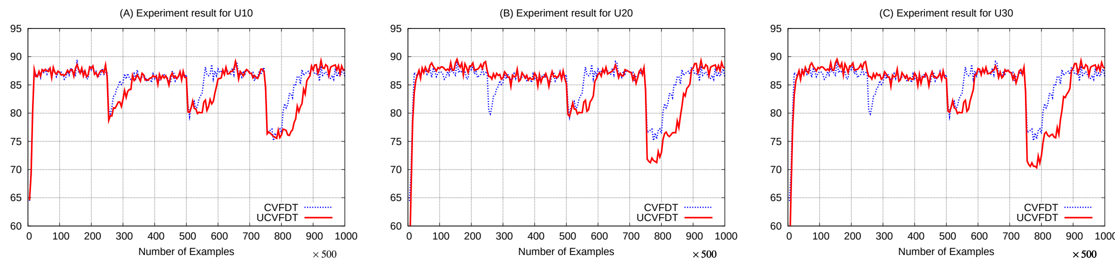

Figure 3: Comparison of CVFDT and UCVFDT on SEA.

Fig.3 gives the detailed view of the experiment result. It can be observed that both CVFDT's accuracy and UCVFDT's accuracy drop badly when data set suffers abrupt concept drift. Because CVFDT learns from certain data, It recover a little better than

UCVFDT. But even uncertainty reaches 20%, the UCVFDT's ability to handle abrupt concept drift is still close to CVFDT.

4.3 Forest Covertype

Forest CoverType is a benchmark problem in the UCI KDD Archive. This problem relates to the actual forest cover type for given observation that was determined from US Forest Service (USFS) Region to Resource Information System (RIS). Forest CoverType includes 581,012 samples with 7 cover type, each sample has 54 attributes including 10 remotely sensed data and 44 cartographic data.

In this group experiments, remotely sensed data is discretized into 5 bins, and were converted into uncertainty with $Uu = 10\%, 15\%$ and 20%. We allow the training examples to arrive one by one to form a data stream and we test the prediction accuracy of the learned model every 5,000 examples throughout the run using the subsequent 5,000 samples as the test data. Experimental results show that average accuracy of CVFDT is 65% while average accuracy of UCVFDT is 65.5%, 65.3% and 63.9% for uncertainty U10,U15 and U20 respectively. Fig.4. shows a detailed view of experimental results. Overall speaking, the accuracy of UCVFDT remains relatively stable. Even when the extent of uncertainty reaches 20%, the accuracy is still quite close to CVFDT over certain data. It shows that UCVFDT is robust against data uncertainty.

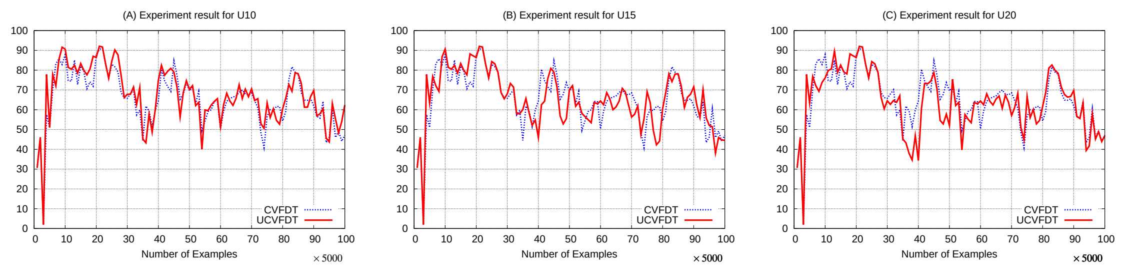

Figure 4: Comparison of CVFDT and UCVFDT on CoverType.

4.4 Analysis of Parameters in UCVFDT

We experiment on moving hyperplane with $d = 30$, $q = 5$, $Uu = U20$, $|S| = 1,000,000$, $p = 5\%$, $Ud = 15$, $K = 5, 10$ and 15.

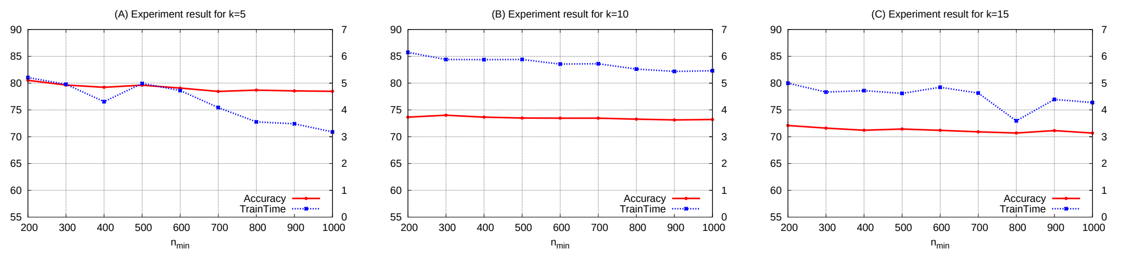

Figure 5: Experiment with n_{min}

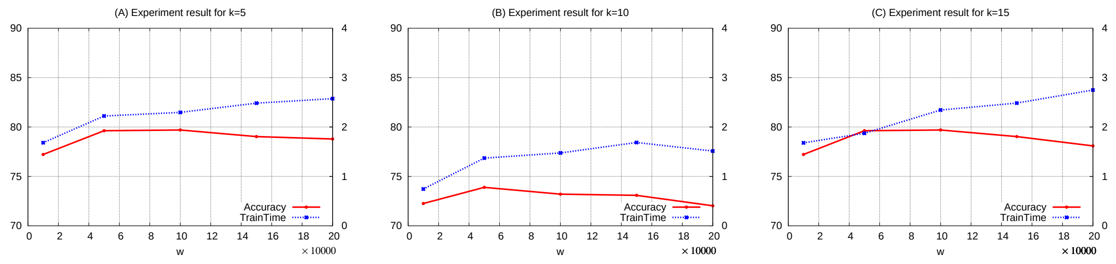

Figure 6: Experiment with w

Parameter n_{min}: We examine how the parameter n_{min} impact UCVFDT's training time and accuracy. Here we set w $= 10,000$. Fig.5 gives the result. In Fig.5, the vertical axis represents classification accuracy and the minor vertical axis represents time lasting of tree building. Fig.5(A), Fig.5(B) and Fig.5(B) gives the experiment result for $K = 5, 10$ and 15. It can be observed from Fig.5 that with increasing of n_{min}, both the classification accuracy and training time is decreasing.

Parameter w: We examine how the size of sliding window impact UCVFDT's training time and accuracy. Here we set $n_{min} = 300$. Fig.6 shows the result. The vertical axis in Fig.6 is the same as in Fig.5. It can be observed from Fig.6 that w can not be neither too small nor too large, or it will lead to a bad performance.

5. CONCLUSION

In this paper, we extended the DTU and CVFDT to a new algorithm UCVFDT. In order to choose the best splitting attribute from uncertain attributes in tree growing, we proposed probabilistic Hoeffding bound and improved the method for computing the uncertain information gain. Our experiments on both synthetic data and real-life data show that even when data set is highly uncertain, UCVFDT still perform excellent. UCVFDT has the ability to handle uncertainty, which makes it more applicable to real-life applications.

Acknowledgments

The authors gratefully acknowledge the support of K. C. Wong Education Foundation, Hong Kong.

References

R. Cheng, D. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data.(SIGMOD'03)*, pages 551–562. ACM New York, NY, USA, 2003.

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

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

J. Gama, R. Fernandes, and R Rocha. Decision trees for mining data streams. *Intell. Data Anal*, 1:23C–45, 2006.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, pages 13–30, 1963.

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.

S. Pan, K. Wu, y. Zhang, and X. Li. Classifier Ensemble for Uncertain Data Stream Classification. In *Proceedings 14th Pacific-Asia Conference on Knowledge Discovery and Data Mining. (PAKDD'09)*, 2009.

B. Qin, Y. Xia, and F. Li. DTU: A Decision Tree for Uncertain Data. In *Advances in Knowledge Discovery and Data Mining.(PAKDD'09)*, pages 4–15, 2009a.

B. Qin, Y. Xia, S. Prabhakar, and Y.C. Tu. A Rule-Based Classification Algorithm for Uncertain Data. In *IEEE International Conference on Data Engineering 2009. (ICDE'09)*, pages 1633–1640. IEEE Computer Society Washington, DC, USA, 2009b.

J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

J. Ren, Lee S., X. Chen, B. Kao, R. Cheng, and Cheung D. Naive Bayes Classification of Uncertain Data. In *Ninth IEEE International Conference on Data Mining. (ICDM'09)*, pages 944–949, 2009.

W.N. Street and Y.S. 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.

S. Tsang, B. Kao, Kevin Y. Yip, Wai-Shing Ho, and Sau Dan Lee. Decision Trees for Uncertain Data. In *IEEE International Conference on Data Engineering 2009. (ICDE'09)*, pages 441–444, 2009.

H. Wang, W. Fan, P.S. 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.

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.

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.