One-Class-Based Uncertain Data Stream Learning

Bo Liu* Yanshan Xiao† Longbing Cao‡ Philip S. Yu§

Abstract

This paper presents a novel approach to one-class-based uncertain data stream learning. Our proposed approach works in three steps. Firstly, we put forward a local kernel-density-based method to generate a bound score for each instance, which refines the location of the corresponding instance. Secondly, we construct an uncertain one-class classifier by incorporating the generated bound score into a one-class SVM-based learning phase. Thirdly, we devise an ensemble classifier, integrated from uncertain one-class classifiers built on the current and historical chunks, to cope with the concept drift involved in the uncertain data stream environment. Our proposed method explicitly handles the uncertainty of the input data and enhances the ability of one-class learning in reducing the sensitivity to noise. Extensive experiments on uncertain data streams demonstrate that our proposed approach can achieve better performance and is highly robust to noise in comparison with state-of-the-art one-class learning method.

1 Introduction

One-class learning on uncertain data streams is a new and challenging research issue. It involves the consolidation of one-class learning [1], data stream mining [2], and uncertain data learning [3]. The problem-solving nature of many real-life applications, such as sensor network, and intrusion detection, can be categorized as one-class-based uncertain data stream learning.

In one-class learning, only one class of samples is labeled in the training phase [1]. The labeled class is typically called the *target class*, while all other samples not in this class are defined as the *non-target class*. The purpose of one-class learning is to build a distinctive classifier to decide whether a test instance belongs to the target class or the non-target class. Such one-class classification problems, often referred to as outlier detection or text classification, have wide real-world applications, typically in intrusion detection, fraud detection, churn management, and Web page classification [4, 5].

The one-class SVM (OC-SVM) [6] is one of the best-known support vector learning methods [7] for one-class classification problems. In OC-SVM, the original data are mapped from the input space into a higher dimensional feature space, in which the inner products of the two vectors can be directly calculated using a kernel function [8]. In the feature space, OC-SVM determines a hyperplane to separate the target class and the origin of the feature space with the maximum margin. The learned hyperplane is thereafter used as a predictive model to classify data into the target or non-target class. In general, OC-SVM outperforms Gaussian-density based linear classifier and Parzen classifier for one-class classification problems [1].

Despite much progress in this area, most of the existing work on one-class classification has not explicitly dealt with the uncertainty of the input data. They are based on an underlying assumption that the training data set does not contain any uncertainty information. However, data in many real-world applications is uncertain in nature. This is because data collection methodologies are only able to capture a certain level of information, making the extracted data incomplete or inaccurate [3]. For example, in environmental monitoring applications, sensor networks typically generate a large amount of uncertain data because of instrument errors, limited accuracy or noise-prone wireless transmission [3]. In other cases, data points extracted in a business may only correspond to business objects that are vaguely specified, which makes the extracted data representation uncertain. This kind of uncertain information, typically ignored in most of the existing one-class learning methods, is critically important for capturing the full picture of the underlying data. It should be considered in the learning phase in order to build a classifier that fully reflects the nature of data.

Recently, advanced data stream technologies have been used to process large amounts of high frequency data [2, 9, 10]. Due to its potential in industry applications, data stream mining has been studied intensively. One characteristic of data streams is the evolution of data concepts (also called concept drift [2]). Meanwhile, a large volume of one-class data, such as sensor network and intrusion detection data, is increasingly collected in a data stream environment which raises the challenging research issue of one-class learning on data streams. An even more challenging issue is to

*QCIS Centre, Faculty of Engineering and IT, University of Technology, Sydney. Email: csbliu@it.uts.edu.au

†QCIS Centre, Faculty of Engineering and IT, University of Technology, Sydney. Email: ysxiao@it.uts.edu.au

‡QCIS Centre, Faculty of Engineering and IT, University of Technology, Sydney. Email: lbcao@it.uts.edu.au

§Department of Computer Science, University of Illinois at Chicago, 851 S. Morgan Street, Chicago, IL 60607-0753. Email: psyu@cs.uic.edu

learn one-class classifiers on uncertain data streams. This is because the presence of uncertain data and concept drift in data streams always makes one-class learning far more difficult than traditional data stream learning methods and uncertain data learning.

In this paper, we address the problem of one-class learning on uncertain data streams. We propose a novel approach, which is called *uncertain one-class learning* (UOCL) framework, which copes with the data uncertainty and the concept drift in one-class-based uncertain data streams. UOCL first assigns a bound score to each instance to estimate the range of uncertainty and then incorporates the uncertainty information into the one-class learning phase. The main contributions of our work are summarized as follows.

1. We propose a local kernel-density-based mechanism to generate a bound score for each instance by investigating its local nearest neighbors in the feature space. By using the generated bound score, we estimate the range of uncertainty such that we can refine the location of the uncertain instance in the learning phase.

2. To cope with data uncertainty, the generated bound score is thereafter incorporated into the OC-SVM learning phase to build an uncertain one-class classifier (UOCC). In this phase, we propose the usage of an interactive framework to mitigate the effect of noise on the one-class classifier. To the best of our knowledge, this is the first time to explicitly handle data uncertainty in one-class uncertain data streams.

3. We introduce the usage of ensemble classifiers which are derived from the current and historical chunks to handle concept drift in uncertain data streams. By using the ensemble classifier, we can capture the evaluation of concept drift in the data stream environment.

4. We conduct extensive experiments to evaluate the performance of our proposed UOCL framework. The statistical results show that UOCL outperforms state-of-the-art one-class learning method in terms of performance and sensitivity to noise added into data.

For clarity, Table 1 lists the notations used in this paper.

The rest of the paper is organized as follows. Section 2 discusses previous work related to this paper. Section 3 introduces the preliminaries of our study. Section 4 presents our proposed approach to one-class-based uncertain data stream learning. Section 5 reports substantial experimental results on mining uncertain data streams. Section 6 concludes the paper and discusses possible directions for future work. Section 7 is the Appendix.

Table 1: Notion and meaning

Notions	Meaning
OC-SVM	
S	training set
$\|S\|$	training set size
$\mathbf{x}_i$	the i^{th} training example
$\mathbf{x}, n$	input space, $n = dim(\mathbf{x})$
$\parallel \mathbf{x}_1 - \mathbf{x}_2 \parallel$	distance between $\mathbf{x}_1$ and $\mathbf{x}_2$
$\mathbf{x}_1 \cdot \mathbf{x}_1$	inner product of two vectors
$\phi(.)$	non-linear mapping function
F	feature space
$\phi(\mathbf{x})$	image of $\mathbf{x}$ in feature space
$K(.,.)$	kernel function
$\mathbf{w}$	normal vector of hyperplane
v	control parameter
ξ_i	slack-variable for i^{th} training sample
ρ	hyperplane: $\rho = \mathbf{w} \cdot \mathbf{x}$
Data stream	
D_t	data stream chunk between $t-1$ and t
$\|D_t\|$	sample size of the chunk t
PD_t	subset of D_t, containing labeled target examples
UD_t	subset of D_t, consisting of unlabeled examples
f_t	the classifier built on the D_t chunk
g_t	weight of classifier f_t
f_E	ensemble classifier
$\triangle \mathbf{x}_i$	noise vector associated with $\mathbf{x}_i$
δ_i	bound score of $\parallel \triangle \mathbf{x}_i \parallel$ such that $\parallel \triangle \mathbf{x}_i \parallel \leq \delta_i$
$\mathbf{x}_i^s$	original uncorrupted input: $\mathbf{x}_i^s = \mathbf{x}_i + \triangle \mathbf{x}_i$
$S_k(\mathbf{x}_i)$	k-nearest neighbors set of instance $\mathbf{x}_i$
$D_{av}\mathbf{x}_i$	average distance between $\mathbf{x}_i$ and its K neighbors
M	distance matrix
ε	threshold
α_i	Lagrange multipliers
Experiment	
σ_i^0	standard deviations of source data along dimension i.
$\mathbf{v}_j$	random vector associated with $\mathbf{x}_j$
$\mathbf{x}_i + \mathbf{v}_j$	corrupted data
η	noise lelve

2 Related work

In this Section, we briefly review previous work related to our study. Since our focus is one-class learning in uncertain data streams, we first review the previous methods for learning uncertain data in Section 2.1, and then briefly introduce one-class learning in Section 2.2.

2.1 Mining Uncertain Data Recently, many advanced techniques have been developed to collect and store large quantities of data. Some records in the data may be interrupted by noise, leading to missing or partially complete data [3]. In such cases, the data objects may be only vaguely specified and are therefore considered uncertain in their representation [3]. The presence of uncertain data has created a great need for uncertain data algorithms [3, 11, 12, 13]. From the application perspective [3], we briefly review the previous work on clustering, classification, frequent pattern mining and outlier detection on uncertain data.

For the clustering methods with uncertain data, most of them typically extend original clustering methods to cope with data uncertainty. The representative methods are introduced as follows. FDBSCAN [14], developed on DBSCAN [15], probabilistically specifies the uncertain distances between objects to find density-based clusters from uncertain

data. FOPTICS [16] introduces a fuzzy distance function to measure the similarity between uncertain data on top of the hierarchical density-based clustering algorithm. UK-means [17], built on the K-means method, assigns an object to the cluster whose representative has the smallest expected distance from the object. Aggarwal [18] proposes the use of density-based approaches to handle error-prone and missing data. The above methods on uncertain data are developed for static uncertain data. Recently, the problem of clustering uncertain data streams has been discussed in [19], which incorporates error statistics and the micro-clustering concept [20] to cope with dynamic uncertain data.

For the classification methods on uncertain data, the standard binary SVM is extended to handle uncertain data [21, 22, 23]. These methods provide a geometric algorithm which optimizes the probabilistic separation between the two classes on both sides of the boundary. Moreover, frequent pattern mining on uncertain data is investigated in [24], in which the probability of an item belonging to a particular transaction is typically modeled. In addition, outlier detection with uncertain data has been studied in [25], which draws multiple samples from the data and computes the fraction of the samples.

Despite much progress on uncertain data mining, most of the previous work has not explicitly dealt with one-class learning on uncertain data. This paper proposes an uncertain one-class learning (UOCL) framework to cope with data uncertainty and concept drift in data streams. In this framework, uncertain one-class classifier (UOCC) extends standard the OC-SVM for uncertain data. Although UOCC is a support vector method, it is different from uncertain binary SVM [21, 22, 23]. Firstly, we propose a local kernel-density-based mechanism to generate a bound score for each instance. Secondly, we simplify our optimization problem (4.9) into a standard QP (quadratic programming) optimization problem (4.11) by considering the characteristic of one-class learning. Thirdly, the UOCC method is embedded into data streams to cope with concept drift.

2.2 One-Class Learning The previous work on one-class learning can be classified into two broad categories. For the methods in the first category [26, 27, 28], they are developed mainly for document-related one-class classification problems. Their main task is to extract negative samples from unlabeled examples (if unlabeled examples are offered) and to construct a binary classifier using target samples and extracted negative samples. The advantage of these methods is that they can well resolve document-related one-class classification problems.

In the second category, OC-SVM and support vector data description (SVDD) are representative methods [6, 1]. Both methods aim to constructe a one-class classifier using only the target class. The advantage of these methods is that they can cope with any one-class classification problem. In this paper, we study one-class learning in this category. Because OC-SVM and SVDD have been proved to have the same decision classifier [1], we briefly introduce OC-SVM [6] as follows.

Suppose the training target class is $S = \{\mathbf{x}_1, x_2, \ldots, \mathbf{x}_{|S|}\}$, where $\mathbf{x}_i \in R^n$. In the input space, OC-SVM aims to determine a hyperplane to separate the target class and the origin of the input space with the maximum margin:

$$\begin{aligned} min \quad & \frac{1}{2} \parallel \mathbf{w} \parallel^2 -\rho + \frac{1}{v \cdot |S|} \sum_{i=1}^{|S|} \xi_i \\ s.t. \quad & \mathbf{w} \cdot \mathbf{x}_i \geq \rho - \xi_i \\ & \xi_i \geq 0, \ \ i = 1, 2, \ldots, |S|, \end{aligned} \tag{2.1}$$

where parameter v is used to tradeoff the sphere volume and the errors $\sum_{i=1}^{|S|} \xi_i$. After resolving problem (2.1) by introducing a Lagrangian function [7], $\mathbf{w}$ and $\rho = \mathbf{w} \cdot \mathbf{x}$ can be obtained.

For a test sample $\mathbf{x}_t$, if

$$\mathbf{w} \cdot \mathbf{x}_t > \rho, \tag{2.2}$$

it is classified into the target class; otherwise, it belongs to the non-target class.

In addition, the kernel method is adopted to offer a more flexible ability of OC-SVM. In this case, the input data are mapped into a feature space, in which the inner product of two vectors $\phi(\mathbf{x})$ and $\phi(\mathbf{x}_i)$ can be calculated by a kernel function $K(\mathbf{x}, \mathbf{x}_i) = \phi(\mathbf{x}) \cdot \phi(\mathbf{x}_i)$. Among a variety of kernel functions, RBF kernel is a typical:

$$K(\mathbf{x}, \mathbf{x}_i) = exp(-\parallel \mathbf{x} - \mathbf{x}_i \parallel_2^2 / 2\sigma^2). \tag{2.3}$$

The original OC-SVM can not cope with the one-class classification problem with uncertain data, which potentially limits its performance. Another disadvantage of OC-SVM is that it is developed on static data and can not deal directly with the concept drift in data streams.

In this paper, we refine the formulation of the original OC-SVM to handle uncertain data, and propose the usage of an ensemble classifier to cope with concept drift in data streams.

3 PRELIMINARY

3.1 Problem Definition Suppose we have a series of data streams D_1, D_2, ..., D_m, in which each $D_t (t = 1, 2, \ldots, m)$, called "chunk", contains the data that arrived between time t_{t-1} and t_t. Here D_c is called the current chunk and the yet-to-come data chunk (denoted as D_{c+1}) is dedicated as the target chunk.

In the current chunk D_c, the labeled examples are put into set PD_c. Other examples, including non-labeled target

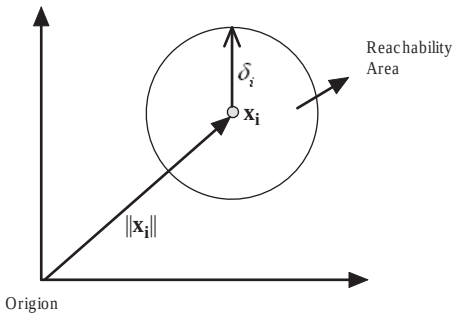


Figure 1: Illustration of reachability area of instance $\mathbf{x}_i$.

class examples and non-target class examples, are put into set UD_c. The objective of one-class data streams is to build a classifier around the labeled target class examples PD_c, and predict the data label in the yet-to-come chunk.

For one-class-based uncertain data streams, we have the following assumption: only instances in the current chunk D_c are accessible, and once the algorithm moves from chunk D_c to chunk D_{c+1}, all instances in chunk $D_t, 1 \leq t \leq c$, are inaccessible; we can only use the models trained from the historical chunks. This is because aggregating historical data always requires extra storage, and most data stream mining algorithms are required to make predictions based on one-scanning of the data streams without referring to historical data.

3.2 Uncertainty Model For the labeled class in the current chunk, that is PD_c, we assume each input data $\mathbf{x}_i$ is subject to an additive noise vector $\triangle\mathbf{x}_i$. In this case, the original uncorrupted input $\mathbf{x}_i^s$ is denoted

$$\mathbf{x}_i^s = \mathbf{x}_i + \triangle\mathbf{x}_i. \tag{3.4}$$

In theory, we can assume $\triangle\mathbf{x}_i$ follows a certain distribution. However, in practice, we may not have any prior knowledge on the noise distribution. Alternatively, the method of bounded and ellipsoidal uncertainties has been investigated in [22, 29]. In this situation, we consider a simple bound score for each instance such that:

$$\| \triangle\mathbf{x}_i \| \leq \delta_i. \tag{3.5}$$

Actually, this setting has a similar effect of assuming $\triangle\mathbf{x}_i$ has a certain distribution. For example, if we assume $\triangle\mathbf{x}_i$ follows a Gaussian noise model, that is

$$p(|\mathbf{x}_i - \mathbf{x}_i^s|) \sim exp^{\frac{\|\mathbf{x}_i - \mathbf{x}_i^s\|^2}{2\sigma^2}},$$

The bound δ_i has a similar influence of the standard deviation σ in the Gaussian noise model. In addition, the squared penalty term $\frac{\|\mathbf{x}_i - \mathbf{x}_i^s\|^2}{2\sigma^2}$ is replaced by a constraint $\| \triangle\mathbf{x}_i \| \leq \delta_i$.

We let $\mathbf{x}_i + \triangle\mathbf{x}_i$ ($\| \triangle\mathbf{x}_i \| \leq \delta_i$) denote the *reachability area* of instance $\mathbf{x}_i$ as illustrated in Figure 1. We then have

$$\begin{aligned} \| \mathbf{x}_i^s \| &= \| \mathbf{x}_i + \triangle\mathbf{x}_i \| \leq \| \mathbf{x}_i \| + \| \triangle\mathbf{x}_i \| \\ &\leq \| \mathbf{x}_i \| + \delta_i \end{aligned} \tag{3.6}$$

In this way, $\mathbf{x}_i^s$ falls in the reachability area of $\mathbf{x}_i$.

By using the bound score for each input sample, we can convert the uncertain one-class learning into standard one-class learning with constraints. We will introduce bound score determination for each instance in Section 4.1 and how to incorporate uncertainty information into the learning phase in Section 4.2.

4 Proposed Approach

In this Section, we provide a detailed description about our proposed approach. Subject to sampling errors or device imperfections, the instance in the one-class-based data streams is always considered uncertain in its representation. Another characteristic of one-class-based data streams is concept drift, which indicates that a user's interest will change as time goes by. These two challenges make one-class-based uncertain data stream learning much more difficult than traditional ones.

To deal with the data uncertainty and concept drift in one-class-based uncertain data stream learning, we propose the uncertain one-class learning (UOCL) framework, as illustrated in Figure 2. The UOCL framework works in three steps:

- In the first step, we generate a bound score for each instance in PD_c based on its local data behavior.
- In the second step, we incorporate this generated bound score into the learning phase to interactively build an uncertain one-class classifier (UOCC).
- In the third step, we integrate the uncertain one-class classifiers derived from the current and historical chunks to predict the data in the target chunk.

In the following, we exhibit the three steps in detail.

For simplicity, we detail the bound score determination and uncertain one-class classifier construction for the current chunk D_c. We can generalize the two operations on other chunks.

4.1 Step One: Bound Score Determination In practice, we are unlikely to know the distribution of $\triangle\mathbf{x}_i$, and it will be difficult to exactly determine the bound score for each instance. One simple solution is to let users set the parameter δ_i themselves. For example, the work in [22] sets a uniform δ for each instance in uncertain binary classification. However, this setting will greatly challenge users, since a big value of δ_i will be meaningless. For example, if we set $\delta_i = \infty$, any

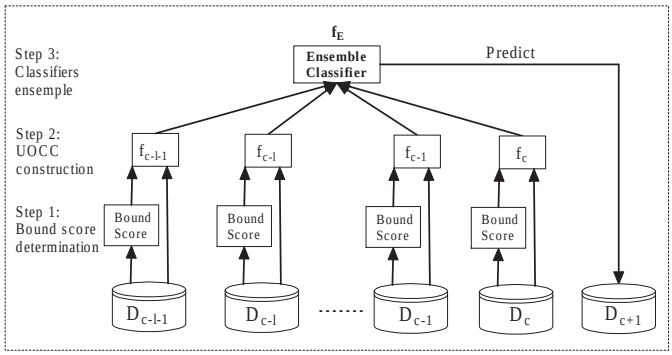


Figure 2: Illustration of one-class-based uncertain data stream learning framework. The ensemble classifier f_E is built on the classifiers ($f_{c-l-1}, f_{c-l}, \ldots, f_c$) derived from l chunks.

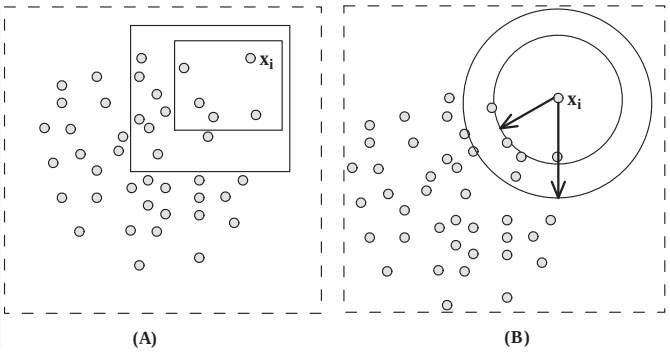


Figure 3: (A): Illustration of four nearest neighbors' range (inner rectangle) and twelve nearest neighbors' range (outer rectangle) of instance $\mathbf{x}_i$. (B): Reachability area of instance $\mathbf{x}_i$ in terms of four nearest neighbors (inner circle) and twelve nearest neighbors (outer circle).

noise $\triangle\mathbf{x}_i$ will satisfy $\| \triangle\mathbf{x}_i \| \leq \infty$, but this setting will not make sense for the solution.

To cope with this challenge, we propose a local kernel-density method to generate a bound score for each instance by examining its local neighbors in a kernel feature space. More specifically, we first identify the k nearest neighbors of instance $\mathbf{x}_i$ in the feature space. The distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ in the feature space is computed as

(4.7)

$$\| \phi(\mathbf{x}_i) - \phi(\mathbf{x}_j) \| = \sqrt{K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_j, \mathbf{x}_j) - 2K(\mathbf{x}_i, \mathbf{x}_j)}.$$

We then calculate the average distance of $\mathbf{x}_i$ to its k nearest neighbors as follows.

$$D_{av}(\mathbf{x}_i) = \frac{1}{|S_{k(\mathbf{x}_i)}|} \sum_{j=1}^{|S_{k(\mathbf{x}_i)}|} \| \phi(\mathbf{x}_j) - \phi(\mathbf{x}_i) \|, \tag{4.8}$$

where $S_{k(\mathbf{x}_i)}$ is the k nearest neighbors set of $\mathbf{x}_i$.

In general, the smaller the $D_{av}(\mathbf{x}_i)$, the greater the density on instance $\mathbf{x}_i$ in feature space.

We then let $\delta_i = D_{av}(\mathbf{x}_i)$. The motivation behind this setting is that: if a normal instance is corrupted by noise such that it resides far from other normal examples, we use this setting to ensure the reachability area of $\mathbf{x}_i$ can reach the regions other normal examples cover. As illustrated in Figure 3, assume $\mathbf{x}_i$ is corrupted by noise, the inner rectangle in the left figure covers the four-nearest neighbors of $\mathbf{x}_i$, and the corresponding reachability area of $\mathbf{x}_i$ (inner circle in the right figure, calculated using four nearest neighbors) can reach the regions other normal examples cover.

Intuitively, a large k will result in a large reachability area of $\mathbf{x}_i$. As illustrated in Figure 3, the reachability area of $\mathbf{x}_i$ calculated using twelve nearest neighbors (outer circle in the right figure) can cover a much greater area than that computed by four nearest neighbors (inner circle in the right figure).

In all, the local kernel-density-based method is given in Algorithm 1.

4.2 Step Two: Uncertain One-Class Classifier Construction

After generating a bound score for each instance in PD_c, the next step is to build an uncertain one-class classifier to cope with uncertain data.

Below, we give a new formulation of OC-SVM by incorporating uncertainty into the learning phase.

4.2.1 Primal Formulation

Based on the bound score for each instance, the formulation of the standard OC-SVM (Optimization problem (2.1)) is extended as follows.

$$\begin{aligned} min \quad & \frac{1}{2} \| \mathbf{w} \|^2 - \rho + \frac{1}{v \cdot |PD_c|} \sum_{i=1}^{|PD_c|} \xi_i \\ s.t. \quad & \mathbf{w} \cdot (\mathbf{x}_i + \triangle\mathbf{x}_i) \geq \rho - \xi_i \\ & \xi_i \geq 0, \ \ \| \triangle\mathbf{x}_i \| \leq \delta_i \\ & i = 1, 2, \ldots, |PD_c|. \end{aligned} \tag{4.9}$$

In problem (4.9), the method of modeling noise lets UOCC less sensitive to the sample corrupted by noise since we can determine a choice of $\triangle\mathbf{x}_i$ to render $\mathbf{x}_i + \triangle\mathbf{x}_i$ far from the decision boundary learned from the OC-SVM. As illustrated in Figure 4, $\mathbf{x}_1$ and $\mathbf{x}_2$ are corrupted samples: assume the dash line is the classifier learned from the standard OC-SVM, after incorporating $\mathbf{x}_1 + \triangle\mathbf{x}_1$ and $\mathbf{x}_2 + \triangle\mathbf{x}_2$ into the learning phase, we obtain the refined boundary denoted by a solid line. In this case, the hyperplane learned from problem (4.9) is less sensitive to uncertain data compared with that derived from the original problem (2.1).

Algorithm 1 Local kernel-density-based method.

Input: PD_c ; // positive set in current chunk D_c
..... k; // number of neighbors
..... $K(.,.)$. // kernel function
Output: δ_i.// bound score

1: Initialize distance matrix $M_{|PD_c|*|PD_c|} = 0$;
2: **for** $i = 1$ to $|PD_c|$ **do**
3: **for** $j = i + 1$ to $|PD_c|$ **do**
4: Calculate $M_{ij} = \| \phi(\mathbf{x}_i) - \phi(\mathbf{x}_j) \|$ via Equation (4.7);
5: Let $M_{ji} = M_{ij}$;
6: **end for**
7: Identify the k nearest neighbors of instance $\mathbf{x}_i$ and put them into set $S_{k(\mathbf{x}_i)}$;
8: **end for**
9: **for** $i = 1$ to $|PD_c|$ **do**
10: Calculate $D_{av}(\mathbf{x}_i)$ via Equation (4.8);
11: Let $\delta_i = D_{av}(\mathbf{x}_i)$;
12: **end for**
13: Return δ_i.

4.2.2 Solution to UOCC To solve the problem (4.9), we use an iterative approach to calculate ρ and $\triangle\mathbf{x}_i$. More specifically, we first fix each $\triangle\mathbf{x}_i$ and solve problem (4.9) to obtain $\overline{\rho}$, and then fix $\overline{\rho}$ to calculate $\triangle\overline{\mathbf{x}}_i$ iteratively. We detail the approach as follows [1].

First of all, we have the following Lemma.

Lemma 1: For a given hyperplane, denoted as $\overline{\rho} = \overline{\mathbf{w}} \cdot \mathbf{x}$, the solution of problem (4.9) over $\triangle\overline{\mathbf{x}}_i$ is

$$\triangle\overline{\mathbf{x}}_i = \delta_i \frac{\overline{\mathbf{w}}}{\| \overline{\mathbf{w}} \|}. \tag{4.10}$$

This Lemma indicates that, for a given $\overline{\mathbf{w}}$, the minimization of problem (4.9) over $\triangle\overline{\mathbf{x}}_i$ is quite straightforward.

We have the Theorem 1 to update $\overline{\mathbf{w}}$ and $\overline{\rho}$ when $\triangle\overline{\mathbf{x}}_i$ is temporally fixed.

Theorem 1: If optimal $\triangle\overline{\mathbf{x}}_i = \delta_i\overline{\mathbf{w}}/\| \overline{\mathbf{w}} \|$ $(i = 1, 2, \ldots, |PD_c|)$ is fixed, the solution of problem (4.9) over $\overline{\mathbf{w}}$ and $\overline{\rho}$ is equivalent to

$$\begin{aligned} min \quad & \frac{1}{2} \| \mathbf{w} \|^2 - \rho + \frac{1}{v \cdot |PD_c|} \sum_{i=1}^{|PD_c|} \xi_i \\ s.t. \quad & \mathbf{w} \cdot (\mathbf{x}_i + \triangle\overline{\mathbf{x}}_i) \geq \rho - \xi_i \\ & \xi_i \geq 0, \\ & i = 1, 2, \ldots, |PD_c|. \end{aligned} \tag{4.11}$$

This Theorem indicates that we can release the constraint $\| \triangle\mathbf{x}_i \| \leq \delta_i$ in problem (4.9) to simplify the optimization.

[1] For the deviation of the Lemma and Theorem 1, please see Section 7 in detail.

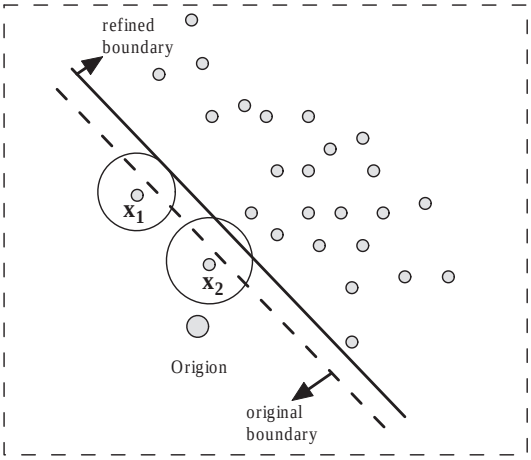


Figure 4: Decision boundaries of the standard OC-SVM and UOCC methods.

We then have Theorem 2 to solve the optimization problem (4.11) as follows.

Theorem 2: By using Lagrangian function [7], the solution of the optimization problem (4.11) is to resolve the following dual problem

$$\begin{aligned} F(\alpha) \quad = \quad & min \, \frac{1}{2} \sum_{i=1}^{|PD_c|} \sum_{j=1}^{|PD_c|} \alpha_i \cdot \alpha_j (\mathbf{x}_i + \triangle\overline{\mathbf{x}}_i) \cdot (\mathbf{x}_j + \triangle\overline{\mathbf{x}}_j) \\ s.t. \quad & 0 \leq \alpha_i \leq \frac{1}{v \cdot |PD_c|} \\ & \sum_{i=1}^{|PD_c|} \alpha_i = 1 \quad i = 1, 2, \ldots, |PD_c|, \end{aligned} \tag{4.12}$$

and

$$\mathbf{w} = \sum_{i=1}^{|PD_c|} \alpha_i \cdot (\mathbf{x}_i + \triangle\overline{\mathbf{x}}_i) \tag{4.13}$$

$$\rho = \mathbf{w} \cdot \mathbf{x}, \tag{4.14}$$

where α_i is the Lagrange multipliers. This theorem indicates that the solution of problem (4.12) is a standard dual problem by treating $\mathbf{x}_i + \triangle\overline{\mathbf{x}}_i$ as a new instance, i.e, $\mathbf{x}' = \mathbf{x}_i + \triangle\overline{\mathbf{x}}_i$.

By referring to the alternating optimization method in [29], we propose the usage of the iterative approach to resolve problem (4.9) in Algorithm 2.

In Algorithm 2, ε is a threshold. Since the value of $F_{val}(t)$ is nonnegative, with the decreasing of $F_{val}(t)$, $|F_{val}(t) - F_{val}(t-1)|/|F_{val}(t-1)|$ will be smaller than a threshold.

4.2.3 Extension to Kernel-Based UOCC In OC-SVM, kernel method always offers a more accurate classifier for one-class classification problem. In such case, each input

Algorithm 2 Uncertain one-class classifier construction

Input: PD_c ; // positive set in current chunk D_c
..... v. // parameter of one-class SVM.
..... δ_i // bound value for each sample.
Output: $\overline{\rho}$.

1: Initialize each $\triangle\overline{\mathbf{x}}_i = 0$;
2: t=0;
3: Initialize $F_{va}(t) = \infty$;
4: **repeat**
5: $t = t + 1$;
6: Fix $\triangle\overline{\mathbf{x}}_i$ for $i = 1, 2, \ldots, |PD_c|$ and solve problem (4.12);
7: Let $F_{va}(t) = F(\alpha)$;
8: Obtain $\alpha_i, i = 1, 2, \ldots, |PD_c|$;
9: Obtain $\overline{\mathbf{w}} = \sum_{i=1}^{PD_c} \alpha_i \cdot (\mathbf{x}_i + \triangle\overline{\mathbf{x}}_i)$ and hyperplane $\overline{\rho} = \overline{\mathbf{w}} \cdot (\mathbf{x})$;
10: Fix $\overline{\mathbf{w}}$ and resolve optimization problem (4.9) to update each $\triangle\overline{\mathbf{x}}_i$ according to Equation (4.10);
11: **until** $|F_{va}(t) - F_{va}(t-1)| < \varepsilon |F_{va}(t-1)|$
12: Return $\overline{\rho} = \overline{\mathbf{w}} \cdot \mathbf{x}_i$.

instance $\mathbf{x}$ is mapped into a feature space via mapping function $\phi(.)$, and the image of $\mathbf{x}$ in feature space is $\phi(\mathbf{x})$. To obtain kernel-based UOCC, we just need to replace $\mathbf{x}_i + \triangle\overline{\mathbf{x}}_i$ with $\phi(\mathbf{x}_i + \triangle\overline{\mathbf{x}}_i)$ in optimization problem (4.9) such that

$$\begin{aligned} min \quad & \frac{1}{2} \parallel \mathbf{w} \parallel^2 - \rho + \frac{1}{v \cdot |D_c|} \sum_{i=1}^{|D_c|} \xi_i \\ s.t. \quad & \mathbf{w} \cdot \phi(\mathbf{x}_i + \triangle\mathbf{x}_i) \geq \rho - \xi_i \\ & \xi_i \geq 0, \ \parallel \triangle\mathbf{x}_i \parallel \leq \delta_i \\ & i = 1, 2, \ldots, |D_c|. \end{aligned} \tag{4.15}$$

The same as solution of problem (4.9), the solution of problem (4.15) consists of two steps. In the first step, fix $\triangle\mathbf{x}_i$ to obtain $\overline{\mathbf{w}}$. In the second step, fix learned $\overline{\mathbf{w}}$ and to optimize $\triangle\overline{\mathbf{x}}_i$. We introduce the two steps as follows.

In step one, we fix $\triangle\mathbf{x}_i$ as a value such that $\parallel \triangle\mathbf{x}_i \parallel \leq \delta_i$, and resolve problem (4.12) by replacing $(\mathbf{x}_i + \triangle\mathbf{x}_i) \cdot (\mathbf{x}_j + \triangle\mathbf{x}_j)$ with $K(\mathbf{x}_i + \triangle\mathbf{x}_i, \mathbf{x}_j + \triangle\mathbf{x}_j)$. We then have

$$\overline{\mathbf{w}} = \sum_{i=1}^{PD_c} \alpha_i \cdot \phi(\mathbf{x}_i + \triangle\mathbf{x}_i). \tag{4.16}$$

In step two, we fix $\overline{\mathbf{w}} = \sum_{i=1}^{PD_c} \alpha_i \cdot \phi(\mathbf{x}_i + \triangle\mathbf{x}_i)$ and resolve problem (4.15) over $\triangle\mathbf{x}_i$, we have

$$\triangle\overline{\mathbf{x}}_i = \delta_i \frac{\overline{\mathbf{w}}}{\parallel \overline{\mathbf{w}} \parallel}. \tag{4.17}$$

Based on the above two steps, we adopt the same iterative approach used in Algorithm 2 to resolve problem (4.15).

After that, we obtain the learned uncertain one-class classifier $f_c = \overline{\mathbf{w}} \cdot \mathbf{x}$ for prediction.

Algorithm 3 Main procedure of UOCL Framework

Input: D_c ; // Current chunk
..... $K(.,.)$ // kernel function
..... f_i $(i = c - l + 1, c - l + 2, \ldots, c)$ // historical classifiers
Output: f_E //ensemble classifier

1: Collect PD_c from D_c;
2: Call Algorithm 1 to calculate δ_i;
3: Call Algorithm 2 using kernel method to obtain f_c;
4: **for** $i = 1,\ i \leq l,\ i++$ **do**
5: Calculate the classifier weight g_i for classifier f_i
6: **end for**
7: Combine f_i $(i = c - l + 1, c - l + 2, \ldots, c)$ to form ensemble classifier f_E;
8: Return f_E.

4.2.4 Computation Complexity Analysis The computation complexity of problem (4.12) is $O(|PD_c|)^2$. The update of $\overline{\mathbf{x}}_i$ in (4.10) just needs linear time, that is $O(|PD_c|)$. Suppose the iterative approach stops after m times iterations. The computation complexity of resolving problems (4.9) and (4.15) is $m \cdot O(|PD_c|)^2 + m \cdot O(|PD_c|)$.

4.3 Step Three: Ensemble Classifiers After building uncertain one-class classifiers on the current and historical chunks, we obtain $f_{c-k+1}, f_{c-k}, \ldots, f_c$. To cope with concept drift in data streams, we combine these k classifiers to form an ensemble classifier f_E to predict instances in the target chunk. By referring to the method in [2], we assign a weight value g_i to each individual classifier such that the ensemble classifier f_E can be quickly adjusted to the users' current interests.

In all, the main framework of the uncertain one-class learning framework is shown in Algorithm 3.

5 Performance evaluation

5.1 Baseline and Metrics In this Section, we will investigate the performance of our proposed UOCL approach on uncertain data streams. For comparison, the original OC-SVM is used as a baseline. Since OC-SVM was developed for static data, we embed it in a data stream environment and use an ensemble classifier to predict yet-to-come data. This baseline is used to show the improvement of our proposed method over the OC-SVM learning method.

The performance of classification systems is typically evaluated in terms of F-measure [30]. The F measure trades off precision p and recall r: $F = 2pr/(r + p)$. From this definition, we know only when both precision and recall are large will the F-measure exhibit large value. For the theoretical basis of F-measure, please refer to [30] for detail.

All the experiments are on a laptop with a 2.8 GHz processor and 3GB DRAM. For both UOCL and OC-SVM, we use Lib-SVM [2] to resolve the standard QP problem. RBF kernel function (2.3) is employed with default parameters.

5.2 Data set Description In the experiments, we use one synthetic data set and two real-world data sets which have been used by other researchers to generate data streams [19, 31]. The basic information of the used data sets is introduced as follows.

1. Synthetic Data: This data set consists of three classes generated using three Gaussian distributions with 10 attributes. We control the mean and standard deviation of each Gaussian distribution such that they won't overlap too much. Specifically, we set the mean of the three Gaussian distributions as 0, 0.5 and 1, the standard deviation of each Gaussian distribution is drawn from the uniform distribution in the range $[0.2, 0.8]$. After that, we generate 15000, 25000 and 20000 samples for three classes.

2. KDD-99 [3]: This data set was collected from the KDD CUP challenge in 1999, and the task is to build predictive models capable of distinguishing between possible intrusions and normal connections. The original data (10% sampling) contain 41 features, 494,020 training and 311,029 test samples, and over 24 intrusion types. In these types, there exist three large size classes, that are Normal, Neptune, and Smurf classes. The three classes dominate the whole data set. We combine training and testing sets together and use the three large size classes to generate a one-class data stream.

3. Forest CoverType[4]: This data set contains 581012 observations and each observation consists of 54 attributes, including 10 quantitative variables, 4 binary wilderness areas and 40 binary soil type variables. Seven classes exist: "Spruce-Fir", "Lodgepole Pine", "Ponderosa Pine", "Cottonwood/Willow", "Aspen", "Douglas-fir", "Krummholz". We use the seven classes to generate one-class data stream.

5.3 Experiment Setting

5.3.1 Data stream Generation

We generate data streams using the above three data sets as follows.

For each data set, we first randomly choose one class, and regard it as target class and treat the other categories as the non-target class. Take the CoverType data set for example: if we choose the "Spruce-Fir" category as the target class, then the non-target class consists of the "Lodgepole Pine", "Ponderosa Pine", "Cottonwood Willow", "Aspen", "Douglas-fir" and "Krummholz". In a sequential time period of the stream environment, we randomly choose a number of examples from the target class and choose examples from the non-target class such that the target class dominates the chosen data. One characteristic of data streams is concept drift. To simulate concept drift in the data stream environment, we employ the following three scenarios: (1) constant concept (CC), (2) regular concept shifting (RCS), and (3) probability concept shifting (PCS).

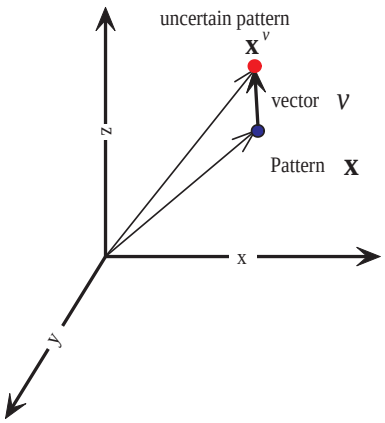


Figure 5: Illustration of the method used to add the noise to a data example: $\mathbf{x}$ is an original data example, $\mathbf{v}$ is a noise vector, $\mathbf{x}^{\mathbf{v}}$ is the new data example with added noise. Here we have $\mathbf{x}^{\mathbf{v}} = \mathbf{x} + \mathbf{v}$.

- In the constant model, users will not change their interests across the whole data stream. In this case, we fix the target class so that it is unchanged all the time. For each data set, we use the biggest class as the normal class and treat the other classes as the non-target class.

- In the regular shifting model, we regularly shift interest by changing the target class from one class to another, after a fixed number of chunks, and we set this chunk number as 5 in the experiments. At the same time, the non-target class is accordingly changed. For example, if the target class changes from "Spruce-Fir" class to "Lodgepole Pine" in the Forest CoverType data set, the changed non-target class will contain "Spruce-Fir", "Ponderosa Pine", "Cottonwood/Willow", "Aspen", "Douglas-fir", "Krummholz".

- In the probability shifting model, we change the target class with a probability (p). If the generated p is larger

[2] Available at http://www.csie.ntu.edu.tw/ cjlin/libsvm/
[3] Available at http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[4] Available at http://archive.ics.uci.edu/ml/datasets/Covertype

than a threshold (0.5), we change the target class from one class into another class.

For the majority of experiments in the paper, we use a regular shifting model and a probability shifting model, because they are closer to real-world models.

Using the above operation, we generate three data streams for each data set, called CC-based, RCS-based, and PCS-based data streams. For fair comparison, both UOCL and OC-SVM use the same number of classifiers to form the ensembles. We omit the details of the algorithm performance with respect to the chunk size ($|D_c|$), and the number of ensemble classifiers (l), because the impact of these factors has been addressed more or less in the stream data mining literature [2, 9, 10]. Instead, we use chunk size N=1000, ensemble size l=10 for all streams. The purpose of fixing these parameters is to fully investigate and compare algorithm performance on uncertain data streams.

For each chunk D_c, we randomly label a majority of target class examples, i.e., approximately 90% target examples, and consider the remaining target class examples and non-target class examples as unlabeled examples. The number of k in the bound score determination step is set as $|PD_c|/10$, and ε is set as 0.15.

5.3.2 Noise Generation We note that the artificial and UCI data sets are deterministic, so we need to model and add uncertainty to these data sets. In the following, we introduce noise addition to PD_c; the same operation can be used on other chunks.

Following the method in the previous work [19],we generate the noise using a Gaussian distribution with zero mean and a standard deviation whose parameter was chosen as follows. For each chunk of data streams, we first compute the standard deviation σ_i^0 of the entire data in PD_c along the ith dimension, and then obtain the standard deviation of the Gaussian noise σ_i randomly from the range $[0, 2\cdot\eta\cdot\sigma_i^0]$. For a generated noise vector, Figure 5 illustrates the basic idea of the method to add the noise vector to a data sample.

Specifically, the standard deviation σ_i^0 of the entire data along the ith dimension is first obtained. In order to model the difference in noise on different dimensions, we define the standard deviation σ_i along the ith dimension, whose value is randomly drawn from the range $[0, 2\cdot\eta\cdot\sigma_i^0]$. Then, for the ith dimension, we add noise from a random distribution with standard deviation σ_i. In this way, a data example $\mathbf{x}_j$ in the target class is added with the noise, which can be presented as a vector

$$\sigma^{\mathbf{x}_j} = [\sigma_1^{\mathbf{x}_j}, \sigma_2^{\mathbf{x}_j}, \cdots, \sigma_{n-1}^{\mathbf{x}_j}, \sigma_n^{\mathbf{x}_j}]. \tag{5.18}$$

Here, n denotes the number of dimensions for a data example $\mathbf{x}_j$, and $\sigma_i^{\mathbf{x}_j}$, $i = 1, \cdots n$ represents the noise added into the ith dimension of the data example.

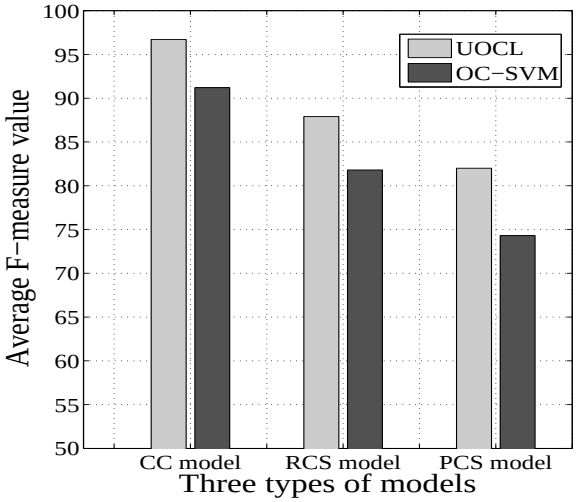


Figure 6: Average F-measure value comparison (Synthetic data, $\eta = 0.5$).

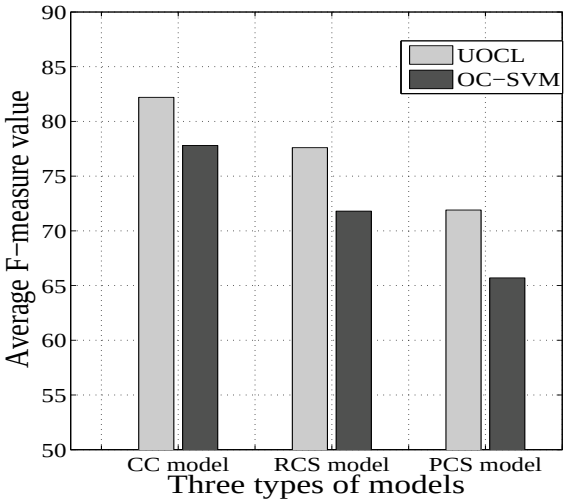


Figure 7: Average F-measure value comparison (KDD-99 data, $\eta = 0.5$).

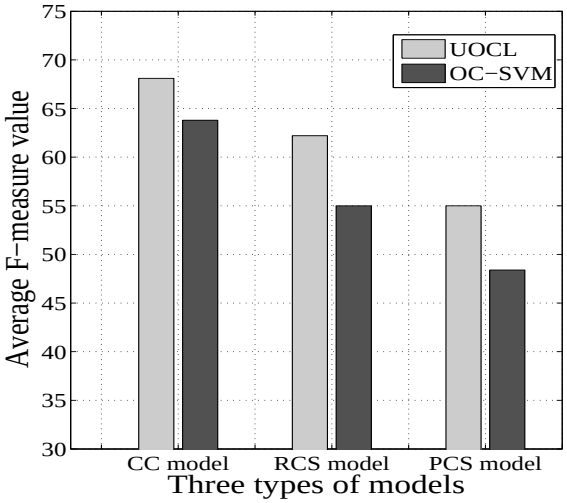


Figure 8: Average F-measure value comparison (Forest CoverType data, $\eta = 0.5$).

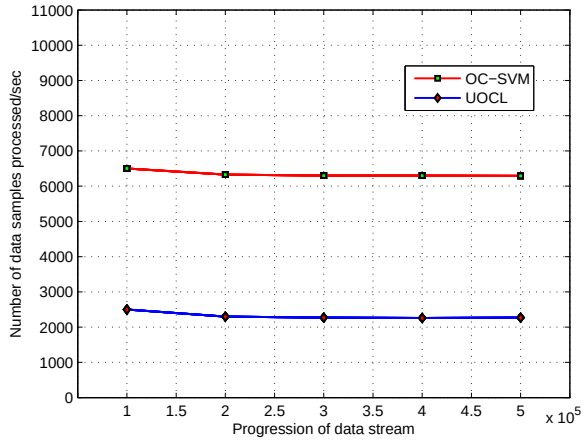


Figure 9: Efficiency comparison (Synthetic data, $\eta = 0.5$).

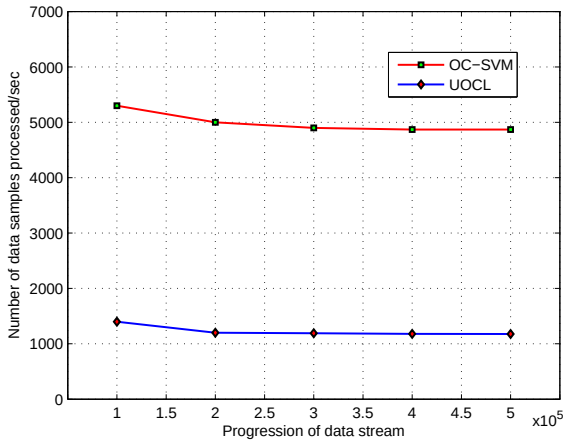


Figure 10: Efficiency comparison (KDD-99 data, $\eta = 0.5$).

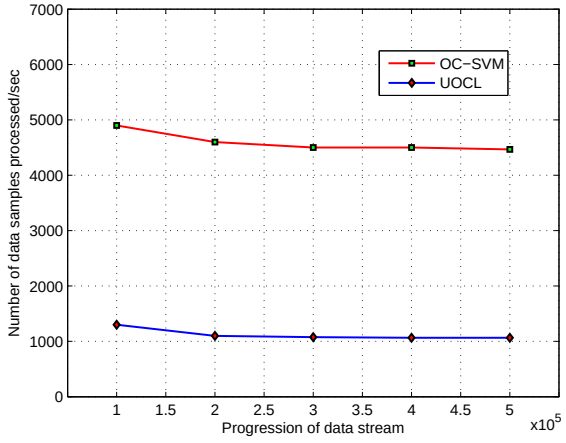


Figure 11: Efficiency comparison (Forest CoverType data, $\eta = 0.5$).

5.4 Performance Comparison We first investigate the performance of UOCL and OC-SVM in terms of CC-based, RCS-based, and PCS-based uncertain data streams, in which the noise level η is set to 0.5. The average F-measure value of forty chunks in data streams is reported from Figures 6 to 8, in which $x-$ axis illustrates the type of the data streams, and $y-$ axis denotes the average F-measure value for each method.

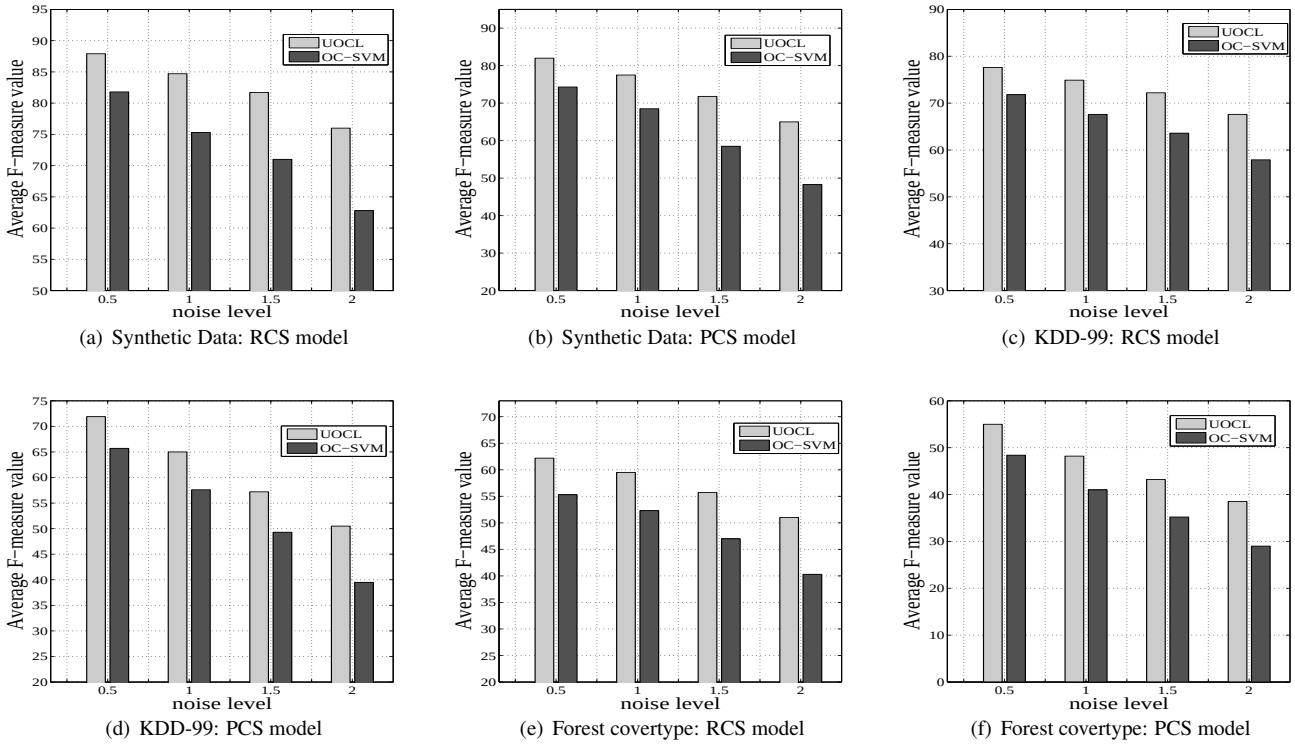
It is clear that in this case, UOCL provides superior performance compared with the standard OC-SVM in terms of CC-based, RCS-based, and PCS-based uncertain data streams. This occurs because UOCL, developed on the OC-SVM, can reduce the effect of the noise on the decision boundary such that we can build a more accurate classifier. Another observation from these Figures is that the F-measure value obtained from CC-based data streams is typically higher than that of RCS-based and PCS-based data streams. This is because CC-based data streams will not involve concept drift so that each chunk shares identical data distribution.

5.5 Efficiency Comparison To test the efficiency of our UOCL method, we refer to the method in [19] and perform UOCL and OC-SVM on PCS-based uncertain data streams in which noise level η is set 0.5. Figures 9 to 11 illustrate the efficiency of both methods on the three data sets. On the $x-$ axis, we illustrate the progression of the data stream in terms of the number of samples, whereas on the $y-$ axis, we illustrate the number of samples processed per second at particular points of the data stream progression. This number is computed by using the average number of samples processed per second in the last 2 seconds.

From these Figures, we find that the original OC-SVM always obtains higher processing speed because it does not mitigate the effect of uncertain data on the classifier construction. Consequently, OC-SVM offers higher processing speed yet inferior performance to UOCL. We further discover that, although UOCL uses an interactive framework to address data uncertainty, the UOCL method is able to process thousands of samples per second in each case. Thus, the UOCL method is not only effective, but is also an efficient one-class method for uncertain data streams.

5.6 Performance Sensitivity on Different Noise Levels So far we have investigated the performance and efficiency of our proposed method. However, it is still interesting to see how the performance of the two methods would be affected under different noise levels.

Taking the RCS-based and PCS-based uncertain data streams for example, we increase the noise level from 0.5 to 2. In Figure 12, we illustrate the variation in effectiveness with increasing error. On the $x-$axis, we illustrate the noise level. On the $y-$axis, we illustrate the average F-measure



(a) Synthetic Data: RCS model (b) Synthetic Data: PCS model (c) KDD-99: RCS model

(d) KDD-99: PCS model (e) Forest covertype: RCS model (f) Forest covertype: PCS model

Figure 12: F-measure value with different noise levels.

value of forty chunks. It is clear that in each case, the F-measure value reduces with the increasing noise level of the underlying data streams. This occurs because when the level of noise increases, the target class potentially becomes less distinguishable from the non-target class. However, we can clearly see that, our UOCL approach can still consistently yield higher F-measure value than OC-SVM. This confirms that, our proposed UOCL can effectively reduce the effect of noise.

We further discover that as the noise level increases, UOCL decreases more slowly than OC-SVM. This is because UOCL has the ability to reduce the effect of noise on the decision boundary by incorporating the uncertainty information into the learning phase. Consequently, UOCL is demonstrated to be robust to noise compared with the standard OC-SVM.

6 Conclusion

In this paper, we propose a new approach to one-class-based uncertain data streams. Our proposed approach first captures the local uncertainty by computing a bound score based on each example's local data behavior, and then builds an uncertain one-class classifier by incorporating the uncertainty information into an OC-SVM-based learning framework. The derived uncertain one-class classifier is thereafter embedded to cope with concept drift involved in the data streams. Extensive experiments have shown that our proposed approach can achieve a better performance and is less sensitive to noise in comparison with state-of-the-art one-class learning method.

In the future, we would like to investigate how to design better mechanisms to generate bound scores based on the data characteristics in a given application domain.

7 Appendix

7.1 Proof of Lemma 1 If the hyperplane is temporally fixed, that is $\overline{\rho} = \overline{\mathbf{w}} \cdot \mathbf{x}$ is given in (4.9), the solution of problem (4.9) equals the minimization of $\sum_{i=1}^{|PD_c|} \xi_i$ over each $\triangle \mathbf{x}_i$.

We assume each noise vector $\triangle \mathbf{x}_i$ corrupts sample $\mathbf{x}_i$ and will not affect other instances. Consequently, $\triangle \mathbf{x}_i$ only has impact on ξ_i. The optimization of $\sum_{i=1}^{|PD_c|} \xi_i$ can be divided to minimize each $\xi_i, i = 1, 2, \ldots, |PD_c|$.

For the minimization of each ξ_i, from the first constraint of problem (4.9), we have

$$\xi_i = \max(0,\ \overline{\rho} - \overline{\mathbf{w}} \cdot \mathbf{x}_i - \overline{\mathbf{w}} \cdot \triangle \mathbf{x}_i). \tag{7.19}$$

By using the Cauchy-Schwarz inequality [32], we have

$$|\overline{\mathbf{w}} \cdot \triangle \mathbf{x}_i| \leq \| \overline{\mathbf{w}} \| \cdot \| \triangle \mathbf{x}_i \| . \tag{7.20}$$

The equality holds only if $\triangle \mathbf{x}_i = c\overline{\mathbf{w}}$, where c is a numerical value. We then have

(7.21)

$$- \parallel \overline{\mathbf{w}} \parallel \cdot \parallel \triangle \mathbf{x}_i \parallel \leq \overline{\mathbf{w}} \cdot \triangle \mathbf{x}_i \leq \parallel \overline{\mathbf{w}} \parallel \cdot \parallel \triangle \mathbf{x}_i \parallel .$$

Since $\parallel \triangle \mathbf{x}_i \parallel \leq \delta_i$, we then have optimal

$$\triangle \overline{\mathbf{x}}_i = \delta_i \frac{\overline{\mathbf{w}}}{\parallel \overline{\mathbf{w}} \parallel}. \tag{7.22}$$

7.2 Proof of Theorem 1 In the optimization problem (4.9), the constraint $\parallel \triangle \mathbf{x}_i \parallel \leq \delta_i$ is used to bound the range of noise vector $\triangle \mathbf{x}_i$. In other words, for a given noise vector $\triangle \mathbf{x}_i$, if $\parallel \triangle \mathbf{x}_i \parallel$ is already less than or equals δ_i, i. e., $\parallel \triangle \mathbf{x}_i \parallel \leq \delta_i$ holds, the constraint $\parallel \triangle \mathbf{x}_i \parallel \leq \delta_i$ in problem (4.9) is useless and will have any impact on the optimization problem (4.9).

For an optimal $\triangle \overline{\mathbf{x}}_i$, $\triangle \overline{\mathbf{x}}_i = \delta_i \frac{\overline{\mathbf{w}}}{\|\overline{\mathbf{w}}\|}$, since

$$\parallel \triangle \overline{\mathbf{x}}_i \parallel = \delta_i \frac{\parallel \overline{\mathbf{w}} \parallel}{\parallel \overline{\mathbf{w}} \parallel} = \delta_i,$$

the constraint $\parallel \triangle \overline{\mathbf{x}}_i \parallel \leq \delta_i$ in problem (4.9) won't have any impacts on this optimization problem.

We can delete the constraint $\parallel \triangle \overline{\mathbf{x}}_i \parallel \leq \delta_i$ from problem (4.9), and then use the optimization problem (4.11) to calculate the updated $\overline{\rho}$.

8 Acknowledgment

This work is sponsored in part by UTS QCIS, Australian Research Council through grants DP1096218, DP0988016, LP100200774 and LP0989721, and US NSF through grants IIS 0905215, DBI-0960443, OISE-0968341 and OIA-0963278.

References

[1] D. M. J. Tax and R. P. W. Duin. Support vector data description. *Machine Learning*, 54(1):45–66, 2004.

[2] H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. *KDD*, pages 226–235, 2003.

[3] C. C. Aggarwal and P. S. Yu. A survey of uncertain data algorithms and applications. *TKDE*, 21(5):609–623, 2009.

[4] R. Perdisci, G. Gu, and W. Lee. Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems. *ICDM*, pages 488–498, 2006.

[5] H. Yu, J. Han, and K. C. C. Chang. Pebl: Web page classification without negative examples. *TKDE*, 16(1):70–81, 2004.

[6] B. Schölkopf, J. Platt, J. S. Taylor, A. J. Smola, and R. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13:1443–1471, 2001.

[7] V. Vapnik. Statistical learning theory. *Springer-Verlag, London, UK*, 1998.

[8] B. Schölkopf and A. J. Smola. Learning with kernels. *Cambridge, MA: MIT Press*, 2002.

[9] W. Street and Y. Kim. A streaming ensemble algorithm (sea) for large-scale classification. *KDD*, pages 377–382, 2001.

[10] J. Gao, W. Fan, J. Han, and P. S. Yu. A general framework for mining concept-drifting data streams with skewed distributions. *SDM*, pages 3–14, 2007.

[11] A. D. Sarma, O. Benjelloun, A. Halevy, and J. Widom. Working models for uncertain data. *ICDE*, pages 163–174, 2006.

[12] D. Burdick, P. M. Deshpande, T. S. Jayram, R. Ramakrishnan, and S. Vaithyanathan. Olap over uncertain and imprecise data. *VLDB*, pages 970–981, 2005.

[13] C. C. Aggarwal. Managing and mining uncertain data. *Springer*, 2009.

[14] H. P. Kriegel and P. Martin. Density-based clustering of uncertain data. *KDD*, pages 672–677, 2005.

[15] M. Ester, H. P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. *KDD*, pages 226–231, 1996.

[16] H. P. Kriegel and M. Pfeifle. Hierarchical density based clustering of uncertain data. *ICDE*, pages 689–692, 2005.

[17] W. Ngai, B. Kao, C. Chui, R. Cheng, M. Chau, and K.Y. Yip. Efficient clustering of uncertain data. *ICDM*, pages 436–445, 2006.

[18] C. C. Aggarwal. On density based transforms for uncertain data mining. *ICDE*, pages 866–875, 2007.

[19] C. C. Aggarwal and P. S. Yu. A framework for clustering uncertain data streams. *ICDE*, pages 150–159, 2008.

[20] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. *VLDB*, pages 81–92, 2003.

[21] P. K. Shivaswamy, C. Bhattacharyya, and A. J. Smola. Second order cone programming approaches for handling missing and uncertain data. *JMLR*, page 1283C1314, 2006.

[22] J. Bi and T. Zhang. Support vector machines with input data uncertainty. *NIPS*, 2004.

[23] S. R. Gunn J. Yang. Exploiting uncertain data in support vector classification. *KES*, pages 148–155, 2007.

[24] C. K. Chui and B. Kao. A decremental approach for mining frequent itemsets from uncertain data. *PAKDD*, pages 64–75, 2008.

[25] C. C. Aggarwal and P. S. Yu. Outlier detection with uncertain data. *SDM*, pages 483–493, 2008.

[26] B. Liu, Y. Dai, X. Li, W. S. Lee, and P. S. Yu. Building text classifiers using positive and unlabeled examples. *ICDM*, pages 179–186, 2003.

[27] G. P. C. Fung, J. X. Yu, H. Lu, and P. S. Yu. Text classification without negative examples revisit. *TKDE*, 18(6):6–20, 2006.

[28] X. Li, P. S. Yu, B. Liu, and S. Ng. Positive unlabeled learning for data stream classification. *SDM*, pages 257–268, 2009.

[29] S. V. Huffel and J. Vandewalle. The total least squares problem: Computational aspects and analysis. *in Frontiers in Applied Mathematics 9. SIAM Press, Philadelphia, PA*, 1991.

[30] J. William and M. Shaw. On the foundation of evaluation. *American Society for Information Science*, 37(5):346–348, 1986.

[31] X. Zhu, X. Wu, and C. Zhang. Vague one-class learning for data streams. *ICDM*, pages 657–666, 2009.

[32] S. S. Dragomir. A survey on cauchy-bunyakovsky-schwarz type discrete inequalities. *Journal of Inequalities in Pure and Applied Mathematics*, 4(3):142, 2003.