One-Class Learning and Concept Summarization for Vaguely Labeled Data Streams*

Xingquan Zhu

Dept. of Computer Science & Engineering, Florida Atlantic University, Boca Raton, FL 33431, USA
Faculty of Eng. & Information Technology, Univ. of Technology, Sydney, NSW 2007, Australia
xqzhu@cse.fau.edu

Wei Ding

Department of Computer Science, University of Massachusetts Boston, Boston, MA 02125, USA
ding@cs.umb.edu

Philip S. Yu

Department of Computer Science, University of Illinois at Chicago, IL 60680, USA
psyu@cs.uic.edu

Chengqi Zhang

Faculty of Eng. & Information Technology, Univ. of Technology, Sydney, NSW 2007, Australia
chengqi@it.uts.edu.au

* A preliminary version of the paper [28], without concept summarization, was published in the Proceedings of the 9th IEEE International Conference on Data mining, Miami, FL, 2009.

Abstract

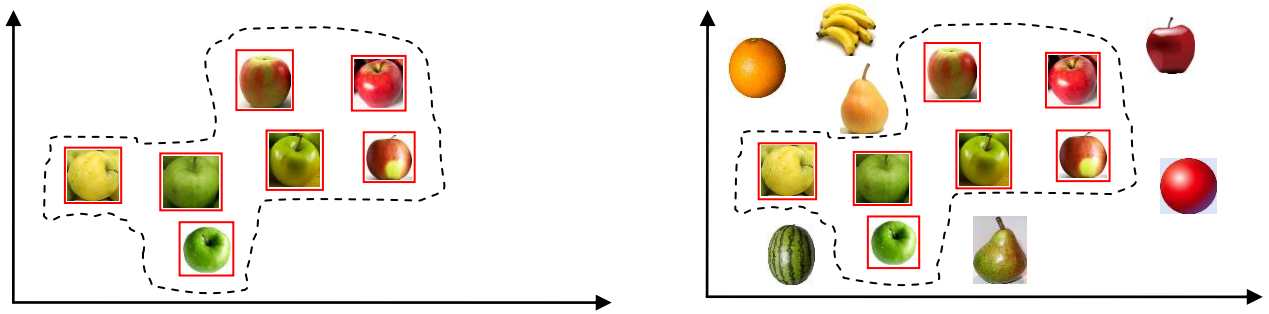
In this paper, we formulate a new research problem of concept learning and summarization for one-class data streams. The main objective is to (1) allow users to label instance groups, instead of single instances, as positive samples for learning, and (2) summarize concepts labeled by users over the whole stream. The employment of the batch-labeling raises serious issues for stream-oriented concept learning and summarization, because a labeled instance group may contain non-positive samples and users may change their labeling interests at any time, so the positive samples labeled by users, over the whole stream, may contain multiple concepts.

To resolve these issues, we propose a One-Class Learning and Summarization (OCLS) system with two major components. In the first component, we propose a Vague One-Class Learning (VOCL) module for concept learning from data streams by using an ensemble of classifiers with instance and classifier level weighting strategies. In the second component, we propose a One-Class Concept Summarization (OCCS) module which uses clustering techniques and a Markov model to summarize concepts labeled by users, with only one scanning of the stream data. Experimental results on synthetic and real-world data streams demonstrate that the proposed VOCL module significantly outperforms its peers for learning concepts from vaguely labeled stream data. The OCCS module is also able to rebuild a high-level summarization for concepts marked by users over the stream.

1. Introduction

One-Class learning, as illustrated in Figure 1, represents a large body of applications [1-14] where only one class of samples[§] (*i.e.*, positive samples) are labeled for training, and the final goal is to predict whether a new instance falls into the same category as the positive examples or not. The niche of the one-class learning stems from the fact that all supervised learning methods require the labeling of the training samples, where finding labeling information for the training data is a time-consuming and cost-intensive process. For many domain applications, such as anomaly detection [1], document classification [2], automatic image annotation [3], authorship verification [4] in scientific writing, transcription factor binding site recognition [52], or other Bioinformatics problems [11], the system can easily collect one class of samples whereas finding (and labeling) instances from other classes are either expensive or time consuming. On the other hand, since the main goal of the learning is to properly identify or recognize samples of interest from a large number of collections, e.g., identifying anomaly [1] or recognizing documents fitting into one particular author's writing style [4], it would be a big advantage if the learning does not require any contrast samples other than the positive instances.

The key challenge of the one-class learning, in comparison to its other peers such as binary- or multi-class classification, lies on the fact that only one class of samples is labeled so general discriminative measures, such as the Gini index [15] or Information Gain [16], are no longer valid for deriving decision logics or modules for predictions. Common solutions are to transform the learning into some well defined optimization problems [6-7], or to treat the learning as a binary classification task if some unlabeled samples are also available [8-10], as illustrated in Figure 1(b). For either case, the quality of the labeled samples plays an important role [17-18], and falsely labeled positive samples will deteriorate learner accuracies significantly, as shown in Figure 2.



(a) One-class learning with positive samples only

(b) One-class learning w. positive and unlabeled samples

Figure 1: A conceptual view of the one-class learning problem for target (apple) recognition. (a) denotes the situation where only positive samples are provided for training, whereas (b) has both positive and unlabeled samples. The final goal of the one-class learning is to find the decision boundaries (the dashed lines) and predict whether a test picture contains an apple or not.

[§] In this paper, instance and sample are equivalent terms; and leaner and classifier are equivalent terms.

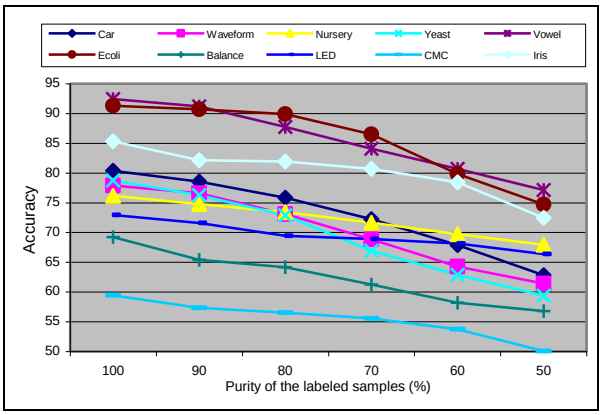


Figure 2: One-class classification accuracy vs. quality of positive samples. The *x*-axis is the percentage of genuine positive instances in the labeled sample set, and the *y*-axis is the accuracy of trained one-class SVM, on 10 datasets from the UCI repository [36]. Results are based on 10 times 10-fold cross validation with each class treated as the positive class one at a time.

With the advent of advanced networking, data collection, storage, and transmission technologies, recent years have witnessed an increasing number of stream-oriented applications [19, 48-51], such as sensor networks [20] or massive audio data streams [48], where data volumes continuously grow and the concepts underneath the data may also evolve or drift rapidly. Under such circumstances, the commonly agreed challenges [21-33] are twofold: (1) continuously growing data volumes; and (2) dynamically drifting [26-27] (or evolving) data concepts. Many solutions exist to handle data streams for classification, clustering [51], and association mining [49], with focus on tackling the above two challenges. From the supervised learning perspective, existing algorithms largely rely on two types of solutions, incremental learning and ensemble learning [21-22], for stream data mining [23-25, 27-33].

Although existing research has significantly advanced the techniques for supervised learning in data stream environments, the main focus, so far, has been limited to tackle the ***data volumes*** and the ***concept drifting*** challenges [47]. By doing so, a labeling process is assumed to accurately label stream data. In a one-class learning scenario, this is equivalent to the setting that an agent (or a user) exists to examine the data and precisely label positive samples (*i.e.* samples of interest to users). For one-class data streams, this becomes a major deficiency or technical barrier. First of all, the continuous nature of the stream data requires users to provide a fast response for labeling. Secondly, due to possible changing of user interests, even carefully labeled positive samples may contain inconsistency or even conflict with each other. Consequently, traditional instance-based labeling is neither effective nor practical for stream-oriented applications. Alternatively, we can allow users to merge samples into groups and label sample groups instead for learning. We call this problem *vague learning*, because such a batch-labeling process is essentially vague and inaccurate in the sense that a labeled group may contain samples not of interest to users and it is not clear which particular instances are genuinely positive.

In addition to the above vague learning challenge, a unique feature for one-class data stream, in comparison with other binary- or multi-class streams [21-33], is that users only label samples interesting to them as positive without paying any attention to the consistency of the concepts behind the labeled positive samples. As the data stream and its labeling process evolve, users' labeling interests may gradually change so the labeled positive samples may contain multiple concepts although they are all labeled as positive. No mechanism, however, is currently available to summarize users' labeling interests. In Figure 3, we illustrate a typical example for sensor network based application, where a user is continuously supervising the sensor stream to monitor some important events (such as raining or change of pressures). To achieve the goal, a user can selectively label some regions, such as region A, B, or C, to indicate that the system is interested in the events associated to the selected region. So data generated from selected regions are regarded as positive samples, and the one-class learning algorithms can be used to monitor or predict regions which share similar patterns as the marked region. Under such a data stream environment, a user labeled region may contain data inconsistent with the underlying event due to reasons such as special environments or malfunctions of a sensor (*i.e.*, vague learning problem). In addition, the user may also frequently switch the region of interest to monitor different type of events. So after monitoring/labeling the data stream for a certain amount of time, it might be needed to reproduce a summarization for user labeled concepts (as shown in the red-dashed rectangle box). More generally, we might need to dynamically summarize user concepts at any particular time, without referring back to the historical stream data. So once the labeling process is done over the whole stream, we are able to summarize the user's interests and restore the concept transferring relationships during the whole labeling process.

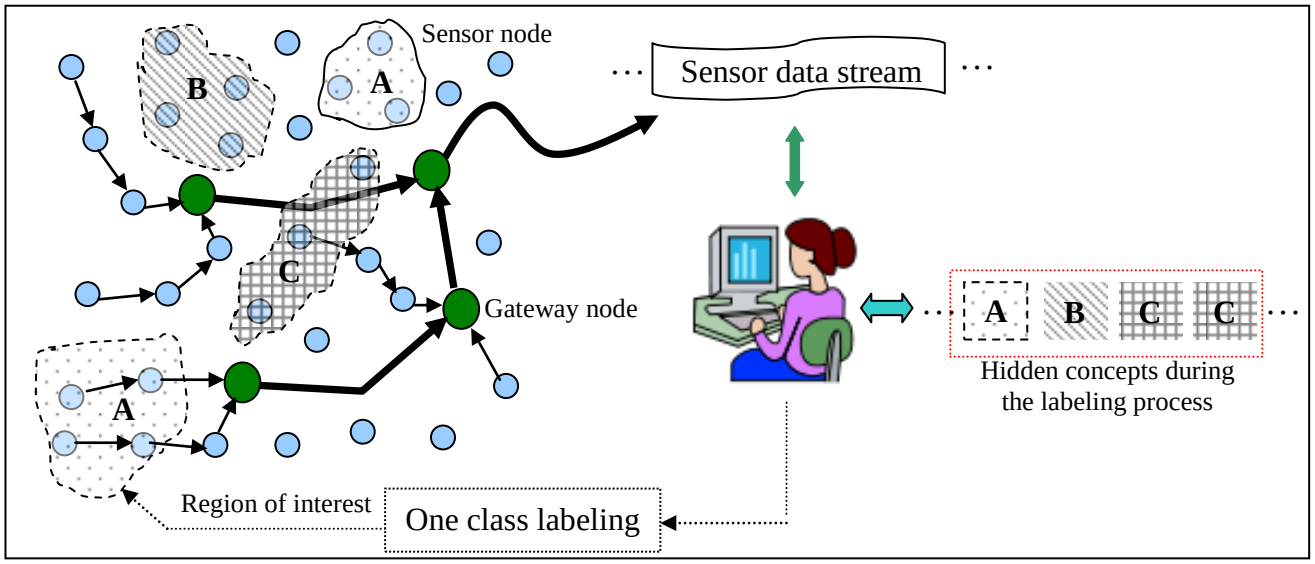


Figure 3: A typical example of a one-class learning based sensor data stream application. Each symbol "A", "B", "C" denotes a region of interesting event (*i.e.*, concept, such as raining, high pressure etc.) sensed by sensors. While sensor data stream flowing continuously, users can select a group of sensors as region of interest (positive class) and monitor similar events (concepts) in the whole data stream. The concepts underneath the user's labeling process thus form a sequence of hidden concepts (the red-dashed rectangle box) which are unknown unless reconstructed or restored explicitly.

In summary, if we consider stream data mining and one-class learning as a whole, the major research challenges are fourfold:

1. **Vague/imprecise positive samples:** Different from traditional one-class learning where positive samples are precisely labeled, in vague one-class learning, a labeled sample set has mixes of instances which may or may not be of interest to users. Effective mechanisms must be developed to differentiate positive samples in order to identify users' genuine interests.
2. **Changing or inconsistent user interests:** During the labeling process, users may frequently change their interests. A learning framework must be able to handle such changes, provide rapid responses, and accurately predict users' current interests.
3. **Increasing data volumes:** Aggregating all examples to build prediction models, like other one-class learning methods do [2-7], is practically infeasible for data streams. One has to devise a stream-oriented one-class learning framework instead.
4. **Scarcity of concept summarization:** Due to the evolving or shifting of the user interests, the positive samples labeled by users may contain multiple concepts. A stream-oriented concept summarization approach is needed to summarize users' interests over the stream.

In this paper, we report our recent progress in addressing the above issues, by using a One-Class Learning and Summarization (OCLS) framework. More specifically, to handle the vague/imprecise positive sample challenge (challenge 1), we assign a weight value to each individual instance to estimate its relevance to users' interests. To handle the changing of user interests (challenge 2), we employ a pair-wise agreement based approach to adjust classifier weights to ensure that the learning can quickly adapt to the users' current interests. By stacking the instance and the classifier level weights together, we build a Vague One-Class Learning (VOCL) module, which naturally solves the data volume challenge (challenge 3). To summarize user concepts (challenge 4), we extract features for each set of samples labeled by users and use a cluster structure map and a concept transfer map to summarize users' interests over the stream.

The remainder of the paper is structured as follows. Section 2 defines the problem, and discusses simple solutions. Section 3 discusses the overall framework of the proposed one-class learning and concept summarization system. Section 4 discusses the vague one-class learning module. Section 5 proposes a solution for concept summarization. Section 6 analyzes the system time complexity. Experimental results are reported in Section 7. We review related work in Section 8 and conclude in Section 9.

2. Problem Definition and Simple Solutions

2.1 Problem Definition

Formally, we assume that a data stream constitutes of samples arriving on a chunk-by-chunk basis, with S_i denoting the i^{th} data chunk (which is often referred to as any chunk or the most recent chunk in the paper) and S_{i-1} denoting the chunk arriving one time step earlier than S_i. Once instances in S_i are collected, users can label a group of instances in S_i as positive samples (denoted by PS_i) and the objective of vague one-class learning is to build a prediction model from historical data chunks …S_{i-3}, S_{i-2}, S_{i-1}, and PS_i, to accurately predict instances in a future chunk S_{i+1}. We call this problem *vague one-class learning*, because we assume samples in PS_i are vaguely or weakly labeled. In our problem setting, we do not intend to find best instance subsets for labeling (*i.e.*, which portion of instances in S_i should be labeled), nor are we interested in identifying falsely labeled positive samples. The purpose of the one-class concept summarization, on the other hand, is to summarize concepts labeled by users in a number of consecutive chunks ……S_{i-3}, S_{i-2}, S_{i-1}, and S_i, where the summary should capture the number of concepts and their transfer relationships over the stream.

For both vague one-class learning and concept summarization, we assume that only instances in the most recent chunk (S_i) are accessible, and once the algorithm moves from chunk S_{i-1} to chunk S_i, all instances in chunk S_{i-1} and its predecessors (…S_{i-3}, S_{i-2}), become inaccessible, except learners or models built from them. The reason we enforce this assumption is because aggregating historical data requires extra storage, and most stream data mining algorithms are required to induce knowledge in one-scanning of the stream data without referring back to the historical data.

2.2 Simple Solutions

Intuitively, the following three methods can be applied to solve the vague one-class learning problem, and no simple solution is available for one-class concept summarization.

Local One-Class Learning (LOCL): LOCL treats data in the current chunk S_i as a static dataset with the training set constituting of labeled samples in S_i (*i.e.*, PS_i). A one-class classifier is trained from PS_i and is used later on to predict samples in S_{i+1}.

Ensemble One-Class Learning (EOCL): EOCL employs an ensemble learning paradigm to combine multiple local one-class classifiers to form a committee for prediction. More specifically, for each data chunk S_i a one-class leaner (L_i) is trained from PS_i (just like LOCL does), and a number of k learners L_{i-k-1},…, L_{i-1}, and L_i trained from PS_{i-k-1},…,PS_{i-1}, and PS_i respectively, are collected to form an ensemble E to predict instances in S_{i+1}.

Filtering One-Class Learning (FOCL): FOCL is motivated by a recent one-class stream data mining effort [8] with an additional data filtering module [17, 37]. For each data chunk S_i and its predecessor chunk S_{i-1}, we first apply cross validation to the labeled portion in S_i (*i.e.*, PS_i) with incorrectly classified positive samples directly excluded from PS_i. In addition, the cross-validation classifiers trained from PS_i are also used to classify unlabeled samples in S_i (*i.e.*, US_i) and all instances in S_{i-1}, with samples, on which the majority classifiers agree to be positive, included into PS_i. Finally, a classifier trained from PS_i is used to predict instances in S_{i+1}. Notice that FOCL has the privilege of accessing instances in both S_i and S_{i-1}, which violates the problem setting defined in Section 2.1 (only instances in S_i are accessible). The purpose of having the relaxation for FOCL is to allow the existing method [8] to be implemented and compared with the solutions we intend to deliver in this paper.

3. VLCS: Vague One-Class Learning and Concept Summarization

In order to address the vague one-class learning and concept summarization challenges, we propose a VLCS system with two major modules, as shown in Figure 4. Given a data stream with samples arriving in a chunk-by-chunk manner, VLCS achieves vague one-class learning by using the VOCL module, and the concept summarization is achieved through the OCCS module.

The process of the vague one-class learning (VOCL) module mainly consists of three layers: stream data, instance weighting, and classifier weighting. At the stream data layer, data are processed in a chunk-by-chunk manner with each chunk S_i containing two subsets PS_i and US_i, where PS_i contains vaguely labeled positive samples (denoted by clusters with blue dots and circles in Figure 4) and US_i contains unlabeled samples in S_i. Once instances in S_i are ready for processing, an instance weighting process is triggered to calculate instance weight values. Instances with their weight values greater than 0 are collected as positive samples (denoted by different sizes of dots or circles in Figure 4). After that, a one-class classifier

is trained from weighted instances, and a number of k classifiers form an ensemble E to predict instances in S_{i+1}. To ensure that predictions can be quickly adjusted to follow users' preferences or interests, a weighting procedure is applied to dynamically tune classifier weight values for predictions.

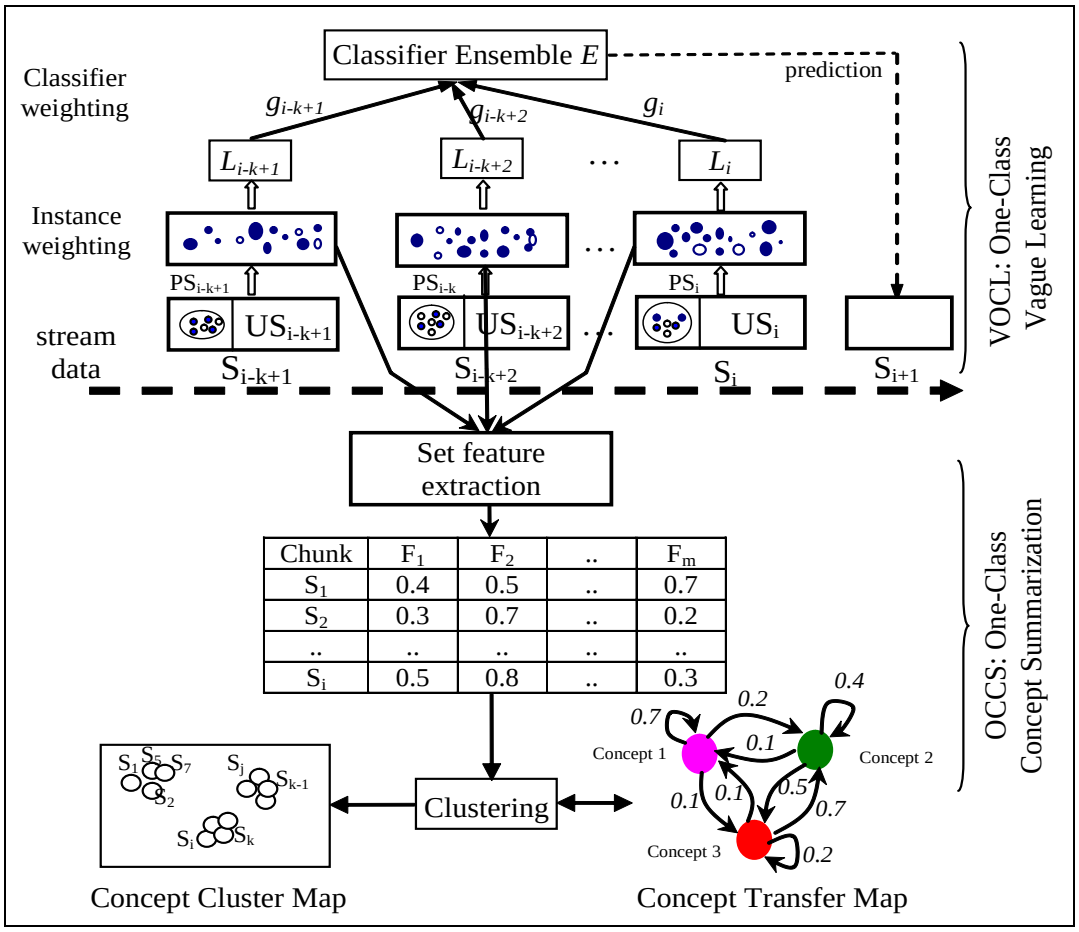


Figure 4: VLCS: Vague one-class learning and concept summarization framework. Two modules included in the framework are VOCL for concept learning and OCCS for concept summarization. Each data chunk S_i constitutes of vaguely labeled instances (PS_i) and unlabeled instance (US_i). A blue dot denotes a genuine positive instance, whereas a blue circle denotes a false positive sample. The size of the circle (or dots) denotes the weight of the instance. For VOCL, the goal is to accurately predict samples in a new chunk S_{i+1}, and for OCCS, the goal is to summarize concepts over the data stream as a cluster map (shown at the bottom-left of the picture) and a concept transfer map (shown at the bottom-right)

One-class concept summarization (OCCS) module directly works on the weighted samples generated from the VOCL module, and its final goal is to restore concepts and their relationships, from the chunks labeled by users, as a concept cluster map and a concept transfer map. For each data chunk S_i, a set feature extraction process is employed to extract features from weighted samples generated from the VOCL module. By doing so, each data chunk S_i is represented as a virtual instance and the whole data streams can be represented as a virtual set with each sample in the set denoting one data chunk. By using generic clustering techniques, we can merge virtual instances as groups, which are regarded as the concept cluster map, and a Markov model is employed to estimate the transfer relationship between concepts over the stream.

Following the overall framework shown in Figure 4, the technical issues remaining are twofold, (1) for VOCL module, we need to properly determine instance weights, employ instance weight values to train one-class classifiers, and assign proper weight values to each classifier to form an ensemble classifier E for prediction, and (2) for OCCS module, we need to transform weighted samples in S_i as a single virtual instance, and discover cluster structure and their transfer relationships from the virtual instance set.

4. VOCL: Vague One-Class Learning

An essential change of the vague one-class learning for data streams is that positive samples are vaguely labeled and may contain non-positive samples. In addition, during the labeling process, users may frequently change their interests so concepts labeled in two consecutive data chunks may be different and need to be differentiated. To solve the problem, we propose to employ a double weighting approach, at both instance and classifier levels, to build an ensemble framework for learning. At instance level, both local and global filtering are considered for instance weight adjustment. At classifier level, a weight value is assigned to each classifier of the ensemble to ensure that learning can quickly adapt to users' interests. In the following sections, we first introduce the overall framework of VOCL, followed by the detailed explanation to each individual component, including instance weight calculation (Section 4.2), classifier weight calculation (Section 4.3), and one-class learning with sample weights (Section 4.4).

VOCL (S, N, k)

Input: Data stream S
N**:** chunk size
k**:** # of classifiers forming the ensemble

Output: Vague one-class prediction model built from S

1. **While**{ new instances arrive}
2. {
3. $S_i \leftarrow$ Collecting instances from S to form a data chunk with N instances
4. $PS_i \leftarrow$ Labeling one or multiple instance groups in S_i as positive
5. $w_x^L \leftarrow$ Local weight calculation (S_i) // Sec. 4.2.1
6. $w_x^G \leftarrow$ Global weight calculation (L_{i-k+1}, L_{i-k+2},…, L_i, S_i) // Sec. 4.2.2
7. $w_x \leftarrow$ Unified weight values // Eq. (3)
8. $L_i \leftarrow$ Training a weighted one-class classifier (S_i, w_x) // Sec. 4.4
9. $g_l \leftarrow$ Classifier weighting (L_{i-k+1}, L_{i-k+2},…, L_i) // Sec. 4.3
10. $E \leftarrow$ Forming weighted classifier ensemble using L_{i-k+1}, L_{i-k+2},…, L_i
11. Calculate E's prediction accuracy on S_{i+1}
12. **For** l from 1 to k-1
13. $L_{i-k+l} \leftarrow L_{i-k+l+1}$
14. **EndFor**
15. i**++;**
16. }
17. Return E and its accuracy

Figure 5: VOCL main procedure

4.1 VOCL Overall Framework

Figure 5 lists the detailed procedures of the VOCL module. Given a data stream S, VOCL takes two parameters N (chunk size) and k (the number of classifiers forming the ensemble) as input. A new data chunk S_i is formed after the collection of N instances. Users can apply any technique (such as k-means clustering) to merge instances in S_i into groups and label one or multiple groups as positive, with instances in the labeled groups forming a positive sample set of S_i (*i.e.*, PS_i). After that, the instance weighting procedures are triggered, as shown on Steps 5 to 9 in Figure 5. A new classifier L_i is trained from a weighted sample set of S_i, and this classifier along with the most recent k-1 classifiers (L_{i-k+1}, L_{i-k+2},…, L_{i-1}) form an ensemble E. The weight of each ensemble member is calculated based on its pair-wise agreement with L_i on unlabeled samples in S_i. From Steps 12 to 14, VOCL updates ensemble E and discards the oldest ensemble member (*i.e.*, L_{i-k+1}) to ensure that the system only maintains the most recent k classifiers. After that, VOCL waits to collect new samples and repeats the while loop if necessary.

4.2 Instance Weighting

The instance weighting procedure is to determine proper weight values for instances in chunk S_i, such that the ones mostly relevant to the users' current interests will receive large weight values, and vice versa. Such a weighting process consists of local weighting and global weighting two parts.

4.1.1 Local Weighting

The local weighting process, as its name suggests, is to determine instance weight values by using samples in local chunk S_i. For this purpose, we assume that majority instances in the vaguely labeled sample set (PS_i) are able to estimate users' interests to some extent, so a one-class classifier built from PS_i can be used to justify whether an instance is of user's interests or not. Based on this assumption, we employ a cross-validation based approach, as shown in Figure 6, to separate instances in PS_i into f non-overlapping subsets. The aggregation of any f-1 subsets forms a training set to build a generic one-class classifier, which is used to classify instances in the excluded subset, say F_j. If a positive instance p in F_j is classified as positive, a weight 1.0 is assigned to p, otherwise p's local weight is set as 0.5. By doing so, the labeled positive instances are treated differently depending on whether the instances are consistent with majority positive samples or not. In addition, because the cross-validation process in Figure 6 trains a total of f classifiers from PS_i, we can use all f classifiers to predict unlabeled instances in US_i, and calculate their weight values as given in Eq. (1).

$$w_u^L = \frac{1}{f} \sum_{j=1, L_j(u)=1}^{f} 1 \ , \quad u \in US_i \tag{1}$$

Local Weighting (S_i)
Input: Current data chunk S_i
Parameters: f: the number of folds for cross-validation
Output: w_x^L, $x \in S_i$, local weight values for all instances in S_i

1. $w_x^L \leftarrow 0$, for every $x \in S_i$
2. Partition labeled instances PS_i into f folds: $S_i = F_1 \cup F_2 \ldots \cup F_f$
3. **For$_a$** each fold F_j, j from 1 to f
4. $\quad \neg F_i \leftarrow S_i \setminus F_i$
5. $\quad L_i \leftarrow$ Train a generic one-class classifier from $\neg F_i$
6. $\quad$ **For$_b$** each instance p in F_i (*i.e.*, positive samples)
7. $\quad\quad$ **If** L_i classifies p as positive
8. $\quad\quad\quad w_p^L \leftarrow 1.0$;
9. $\quad\quad$ **Else**
10. $\quad\quad\quad w_p^L \leftarrow 0.5$;
11. $\quad$ **EndFor$_b$**
12. $\quad$ **For$_c$** each instance u in US_i (*i.e.*, unlabeled samples)
13. $\quad\quad$ **If** L_i classifies u as positive
14. $\quad\quad\quad w_u^L \leftarrow w_u^L + 1$;
15. $\quad$ **EndFor$_c$**
16. **EndFor$_a$**
17. $w_u^L \leftarrow$ Normalize weight values for unlabeled set US_i
18. **Return** (w_x^L, $x \in S_i$)

Figure 6: Local instance weighting procedure

4.1.2 Global Weighting

The purpose of global weighting is to determine a weight value for both positive and unlabeled instances in S_i, by using a number of classifiers trained from the chunks preceding to the current chunk S_i. More specifically, given data chunk S_i and k-1 classifiers, L_{j-k+1}, L_{j-k+2}, …, L_{j-1}, trained from PS_{i-k+1}, PS_{i-k+2}, …, PS_{i-1}, the global weight of an instance x in S_i is the percentage of classifiers predicting x as positive, as given by Eq. (2).

$$w_x^G = \frac{1}{k-1} \sum_{j=i-k+1, L_j(x)=1}^{i-1} 1 \quad , \quad x \in S_i \tag{2}$$

To combine local and global weights to form a single measure, we calculate the summation of local and global weight values and further divide this value by the difference between two weights, as defined in Eq. (3), where *a* and *b* are the Laplace smoothing parameters [34], which control the range of the final output (in our experiments, we set *a* and *b* both equal to 0.5, which gives unified weight values between [1,5]). Obviously, the measure defined by Eq.(3) favors instances with large local and global values, and will return a maximum value for instances with maximum local and global weights.

$$w_x = \begin{cases} \dfrac{w_x^L + w_x^G + a}{\| w_x^L - w_x^G \| + b}, & If \ w_x^L + w_x^G > 0 \\ 0 & Otherwise \end{cases} \tag{3}$$

4.3 Classifier Weighting

Assuming a number of *k* one-class classifiers, L_{i-k+1}, L_{i-k+2}, .., L_i, are forming an ensemble *E* to predict instances in S_{i+1}, an important issue is to assign a proper weight value to each individual classifier in *E*, such that the final predictions can be quickly adjusted to the users' current interests. Traditionally, this problem is solved by setting each classifier's weight as its error rate (or accuracy) on an evaluation set which shares the same distribution as the test set [24-25, 27, 31]. For vaguely labeled data streams, this approach has three disadvantages: (1) in one-class learning paradigm, the number of labeled positive samples is very limited; (2) finding an evaluation set sharing the same distribution as the test set (*i.e.*, S_{i+1}) is essentially difficult, because user interests may change without any indications; and (3) the most recently labeled set PS_i cannot be used as an evaluation set because its instances are vaguely labeled, whereas in other stream data environments the most recent training set can be used as the evaluation set. To address these issues, we propose to use a pair-wise agreement between a classifier L_l and the most recent classifier L_i to assign a weight value for L_l, as defined by Eq.(4), where the most recent classifier L_i will receive the largest weight value 1.

$$g_\ell = \frac{1}{|US_i|} \sum_{j=1, x_j \in US_i, L_\ell(x_j)=L_i(x_j)}^{|US_i|} 1 \tag{4}$$

4.4 One-Class Learning with Sample Weights

4.4.1 One-Class SVM

One-class SVM is a special type of support vector machines, where learning intends to find a hyper-sphere enclosing all positive samples [6] or hyper-planes separating positive examples from the origin with the maximum margin [7].

Given a number of *n* positive samples, $X=\{x_1, x_2, \ldots, x_n\}$ the objective function of the Schölkopf OC-SVM model [7] is to discover the hyper-plane, determined by $W^T \cdot X = \rho$, that separates the positive samples from the origin with the maximal margin. Because many non-linear problems may be linearly separable after proper transformations, kernel transformations $\phi()$ are normally employed to transform an input example from one space to another, which gives the hyper-plane denoted by $W^T \cdot \phi(X) = \rho$. This objective is defined by the convex problem given in Eq. (5), where *W* is orthogonal to the determined hyper-plane, ν is the fraction of positive samples not separated from the origin by the determined hyper-plane. x_i is the i^{th} positive sample and ξ_i is a slack variable which defines the "penalty" if a sample is not separated from the original.

$$\begin{aligned} &\min_{W, \xi_1, \cdots, \xi_n, \rho} \frac{1}{2}\|W\|^2 - \rho + \frac{1}{\nu \cdot n} \sum_{i=1}^{n} \xi_i \\ &s.t. \quad W^T \cdot \phi(x_i) \geq \rho - \xi_i \\ &\xi_i \geq 0, \quad l \in [1,..,n] \end{aligned} \tag{5}$$

In Eq. (1), $\phi(x_i)$ defines a kernel function which transforms an input example from one space to another. Numerous kernels, such as linear, polynomial, and Gaussian kernels, exist for such purposes. Eq. (6) gives the Gaussian kernel (also called Radial Basis Function), one of the most popularly used kernels in SVM.

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j) = \exp(-\gamma \left\| x_i - x_j \right\|^2), \quad \gamma > 0 \tag{6}$$

The solutions to Eq. (5) will return weight values W, which determine the hyper-plane. If a new instance x_t is classified as a positive example if it satisfies $W^T \cdot \phi(x_t) \geq \rho$, otherwise, it is regarded as a non-positive sample.

4.4.2. Weighted Sampling for One-Class Learning

Traditional one-class learners, such as one-class SVM, assume that all training samples are equally important, and therefore cannot directly accommodate instance weight values for learning. To solve the problem, we consider that the weight value of each instance denotes the density of the given instance, so we can use weight values to change training sample distributions, such that an instance with a larger weight value will have a better chance in influencing the formulation of the decisions, whereas a zero weight instance will not participate in the training at all. This objective can be achieved by employing a sampling technique to form a new training set with its distribution biased towards instances with large weight values. After that, we can train a one-class classifier from the sampled set for prediction.

Given a training set S with n positive instances, denoting (x_i, w_i) an instance x_i with weight value w_i, a straightforward way for weight-based sampling is to employ sampling-with-replacement to sequentially examine each instance x_i from S and include x_i into a new set S' with probability given in Eq. (7). Such a sampling-with-replacement approach, however, cannot guarantee that instances are drawn independently from S to form a new set S', so it cannot ensure that the sample set S' is formed from the same distribution as S.

$$p(x_i, w_i) = \frac{w_i}{\sum_{l=1,\,(x_l, w_l) \in S}^{n} w_l} \tag{7}$$

$$c = \max_{x \in S} \frac{p(x, w)}{g(x)} \tag{8}$$

To produce a sample set which exactly suits our needs, the Rejection Sampling [38] from statistics provides a solution to ensure that instances are independently sampled from the given distribution. In short, suppose we need to sample from a target distribution $f(x)$ to form another distribution following $f(x)$. Assume further that we have a second distribution $g(x)$ (*e.g.*, a uniform distribution) from which we have a reasonable method of independent sampling. If there is a constant c | $c \cdot g(x) \geq f(x)$ $\forall x$, then the pseudo-code given in Figure 5 will result in a sample set following distribution $f(x)$ with instances independently sampled from the given distribution.

Rejection-Sampling (S, m)
Input: An n-instance set S with weight values (x_i, w_i), $i \in [1,n]$
m: The size of the sampled set
Output: S', a sampled set following the same distribution as S.

1. $S' \leftarrow \varnothing$
2. $p(x_i, w_i) \leftarrow$ Calculate pdf for all instances x_i in S // Eq. (7)
3. $g(x) \sim$ Uniform {1,2, ..., n}. Choose $g(x)$ to be a discrete uniform distribution with n elements.
4. $c \leftarrow$ find constant ensures that $c \cdot g(x) \geq f(x)$ $\forall x$ // Eq. (8)
5. **While** {$|S'| < m$}
6. $l \leftarrow$ generate a sample from the $g(x) \sim$ Uniform {1,2, ..., n}
7. $\omega \leftarrow$ generate another sample ω from Uniform[0, 1]
8. $S' \leftarrow S' \cup \{x_l\}$ if $\omega \leq \dfrac{p(x_l, w_l)}{c \times g(x)}$
9. **Endwhile**
10. **Return** (S')

Figure 7: Rejection sampling for generating new sample set w.r.t. the sample weights

Given a sample set S, we can employ the sampling mechanism in Figure 7 to build a number of sets S'_1, S'_2, …, S'_π, each of which is of the same size as S. Denoting L_l a one-class learner trained from S'_l, the final prediction on an instance x_t is based on the majority voting of all classifiers, as shown in Eq. (9) where π denotes the number of sets sampled from S.

The merits of the above sampling based one-class learning approach are threefold: (1) rejection sampling independently draws instances to form a new distribution, w.r.t. instances' weight values. It is inherently superior to other sampling mechanisms like sampling with/without replacement, because instances are not independently drawn from the latter approach [39]; (2) the final predictions, which are the voting of the classifiers trained from a number of sampled sets, are typically more accurate than what a single classifier can offer; and (3) the weighted sampling approach can be applied to any one-class learner to take instance weight values into consideration for learning.

$$L(x_t) = \arg\max_{y\in\{1,-1\}} \sum_{l=1, L_l(x_t)=y}^{\pi} 1 \tag{9}$$

5. OCCS: One-Class Concept Summarization

As we explained in Section 1, for one-class data streams, the users simply label all samples as positive without any indication on the number of concepts and their relationships. The main objective of the OCCS module is to dynamically summarize the concepts labeled by users in the stream. To achieve the goal, we propose to employ clustering based techniques to merge data chunks into a number of groups, each of which denotes a concept. Following the clustering process, two types of summaries, concept cluster map and concept transfer map, are provided. The concept cluster map directly employs the cluster structure to visually demonstrate concept-chunk relationships in the stream. For concept transfer map, we build a Markov model to capture concept transfer patterns during the users' labeling process. In the following subsections, we first propose a set feature extraction technique for concept clustering, followed by a process of using Markov model to generate a concept transfer map from the stream data.

5.1 Concept Summarization Using Set Feature Based Clustering

The set feature based clustering process includes two major steps: (1) extract features from each chunk (*i.e.*, set) and represent all chunks as a virtual sample set; and (2) generate cluster structures from the virtual set. To extract feature for each chunk, we propose to transform original feature values in each chunk into some histogram format, such that each chunk S_i can be represented by using histogram based features. More specifically, assume the given of a numerical feature v with all feature values falling in the range $[v_{min}, v_{max}]$. The feature histogram for v is constructed by separating all values between $[v_{min}, v_{max}]$ into a number of B bins. The value in each bin b, $b\in[1, B]$ denotes the percentage of labeled instance in a chunk S_i with their value in feature v falling in the range between $v_{min}+(b-1)\times(v_{max}-v_{min})/B$ and $v_{min}+b\times(v_{max}-v_{min})/B$. If feature v is a categorical feature with the domain $v\in\{v_1, v_2,\ldots, v_{|v|}\}$, the feature histogram for v is denoted by a total of $|v|$ bins, with the value in each bin b, $b\in[1, |v|]$, denoting the percentage of labeled instances in S_i with their value in feature v equal to v_b.

The above set feature extraction process only considers labeled samples in each chunk S_i, whereas the labeled samples in S_i may contain non-positive samples and deteriorate the accuracy for concept clustering. Recall that VOCL learning process in Section 4 actually generates a weight value for each sample in chunk S_i, with the weight denoting each sample's relevance to the users' current interests. Consequently, the set feature extraction procedure will also take the sample weights into consideration for set feature extraction. Formally, given a set of instances x_j in a chunk S_i, each of which is associated to a weight value $w_{i,j}$, we take the weight of each instance as its density and calculate the feature histogram for bin b of the feature v using Eq. (10), where x_j^v denotes the value of the feature v of instance x_j and B_b denotes the range of feature values corresponding to the b^{th} bin of the feature histogram for v.

$$\Gamma_b^v = \frac{\sum_{x_j\in S_i, x_j^v\in B_b} w_{i,j}}{\sum_{x_j\in S_i} w_{i,j}} \tag{10}$$

Take the toy chunk in Figure 8 as an example, where five instances each have two attributes (one numerical and one categorical) and one weight value. Assume the minimum and the maximum values of feature v_1 are 0 and 10, each numerical feature is represented by B=5 bins, and the domain of the categorical attribute v_2 is {T, F}, the set features

extracted from the chunk S_i is shown at the bottom of Figure 8. In summary, the set features extracted from the above process represent the histogram of each individual feature in the original feature space. Although such a representation considers each feature independently and the interactions of the feature values in the original set S_i are not longer preserved, in many applications such as sensor networks the information collected for each feature (*i.e.*, each sensor) is largely independent of each other. As a result, the whole process considers the statistical information of the whole set with respect to each individual feature to produce concise representation for each labeled chunk.

Following the above set feature extraction process, the whole data stream can be represented by a virtual set with each instance in the set denoting one individual chunk. Traditional clustering techniques, such as k-means or spectral clustering methods, can be directly applied to discover cluster structure from the data. Because each chunk S_i is transformed as a single instance with much smaller storage space, the original samples in S_i are no longer needed in the succeeding process, so only one scanning is needed to generate a cluster map from the stream data.

Chunk S_i

Instance ID	v_1	v_2	Instance weight
1	2.5	T	3.0
2	5	T	1.0
3	3	F	4.0
4	7	F	2.0
5	6.2	T	0.0

Virtual Instance	Γ_1^1 [0,2]	Γ_1^2 (2,4]	Γ_1^3 (4,6]	Γ_1^4 (6,8]	Γ_1^5 (8,10]	Γ_2^1 T	Γ_2^2 F
Γ_{S_i}	0	0.7	0.1	0.2	0	0.4	0.6

Figure 8: A toy data chunk S_x with five instances, each of which has a numerical feature v_1, a categorical feature v_2, and a weight value. The virtual instance and the set features extracted from the toy set are shown at bottom of the picture.

5.2 Concept Summarization Using Markov Model

The clustering based concept summarization approach introduced in Section 5.1 provides a cluster map to summarize the concepts labeled by users. An inherent disadvantage of such a summarization approach is that the temporal correlation of the concepts during the whole labeling process was discarded. In this subsection, we propose to employ Markov model to build a state transfer map to capture detailed temporal correlations between concepts.

During the whole labeling process, assume the genuine concept behind the users is denoted by a random variable C, and the labeled samples in each data chunk represent one single concept. When taking the whole data stream chunks as a sequence, we can collect a set of random variables C_1, C_2, C_3,…C_n, where n denotes the total number of chunks. Let's assume further that the clustering process in Section 5.1 outputs a total number of k concepts (*i.e.* clusters), denoted by c_1, c_2, …, c_n, which form a countable set $c=\{c_1,c_2,...,c_k\}$ denoting the concept space of the users' interests. It is clear that the concepts labeled in consecutive chunks are rarely independent of each others but rather share some temporal correlations. For any chunk S_i, the probability for users to choose a particular concept c_x, $c_x \in c$, is given by $P(C_i = c_x)$. Given a number of n consecutive chunks and labeled concepts, $C_1 = c_{x_1}$, $C_2 = c_{x_2}$, $C_3 = c_{x_3}$, …, $C_n = c_{x_n}$ where $x_l \in [1, k]$ and $l \in [1, n]$, the joint probability of the concepts, given the observed sequence S, are given in Eq. (11)

$$P(C \mid S) = P(C_1 = c_{x_1}, C_2 = c_{x_2}, \cdots, C_n = c_{x_n}) \tag{11}$$

Using probability product rule, the above joint probability is equivalent to Eq. (12) where $P(C_i \mid C_j)$ is a shorthand of $P(C_i = c_{x_i} \mid C_j = c_{x_j})$.

$$\begin{aligned} P(C \mid S) &= P(C_1 = c_{x_1}, C_2 = c_{x_2}, \cdots, C_n = c_{x_n}) = P(C_n \mid C_1, C_2, ..., C_{n-1}) P(C_1, C_2, ..., C_{n-1}) \\ &= P(C_n \mid C_1, ..., C_{n-1}) P(C_{n-1} \mid C_1, ..., C_{n-2}), ..., P(C_2 \mid C_1) P(C_1) \end{aligned} \tag{12}$$

Assume that a user's labeling process follows the first order Markov chain property [40] (*i.e.*, the labeled concept in the current chunk S_i depends on the concept of the previous chunk S_{i-1} only), the Eq.(12) can be simplified as

$$P(C \mid S) = P(C_n \mid C_{n-1})P(C_{n-1} \mid C_{n-2}),..., P(C_2 \mid C_1)P(C_1) \tag{13}$$

According to the Maximum Likelihood principle [41], the cluster structures $c=\{c_1,c_2,...,c_k\}$ should be selected such that the log likelihood denoted by Eq. (14) can reach a minimum,

$$\arg\min_{k;c=\{c_1,c_2,...c_k\}} \lg P(C \mid S) = \arg\min_{k;c=\{c_1,c_2,...c_k\}} \left[\lg P(C_n \mid C_{n-1}) + \lg P(C_{n-1} \mid C_{n-2}) + \cdots + \lg P(C_1)\right] \tag{14}$$

To solve Eq. (14), we must explicitly calculate the conditional probability $P(C_i \mid C_{i-1}) = P(C_i = c_{x_i} \mid C_{i-1} = c_{x_{i-1}})$. Intuitively, this probability can be roughly estimated by discarding the data chunk ID and directly calculate the concept transfer probability as $P(c_{x_i} \mid c_{x_{i-1}})$, $c_x \in c$, over the whole observed concept sequence *S*. Such calculation is, however, inaccurate, mainly because that $P(c_{x_i} \mid c_{x_{i-1}})$ does not take the probability of each chunk S_i to have the concept c_{x_i} into consideration, so it may not be able to accurately estimate $P(C_i = c_{x_i} \mid C_{i-1} = c_{x_{i-1}})$. To solve the problem, we consider a generative graph model as shown in Figure 9, where the conditional probability $P(C_i = c_{x_i} \mid C_{i-1} = c_{x_{i-1}})$ is determined by three independent factors, including (1) the probability that the concept in chunk S_{i-1} is $c_{x_{i-1}}$, denoted by $P(C_{i-1} = c_{x_{i-1}})$, (2) the probability that a concept transfers from $c_{x_{i-1}}$ to c_{x_i}, denoted by $P(c_{x_i} \mid c_{x_{i-1}})$, and (3) the probability that the concept in chunk S_i is c_{x_i}, denoted by $P(C_i = c_{x_i})$.

Following the graphical model in Figure 9, the calculation of the probability $P(C_i \mid C_{i-1})$ is explicitly given in Eq. (15). To estimate the probability of the chunk S_i to have a particular concept c_{x_i}, *i.e.*, $P(C_i = c_{x_i})$, we can use the likelihood of the chunk S_i belonging to the concept c_{x_i}, which is can be directly derived from the clustering results.

$$P(C_i \mid C_{i-1}) = P(C_{i-1} = c_{x_{i-1}})P(c_{x_i} \mid c_{x_{i-1}})P(C_i = c_{x_i}) \tag{15}$$

The benefit of using Eq. (14) for concept summarization is twofold. (1) the whole process searches the clustering space to find the cluster structures such that the likelihood of the concept sequences S generated from the stream can be maximized, so the clustering and the summarization are seamlessly integrated to achieve an optimization goal, and (2) the concept transfer matrix $P(c_{x_i} \mid c_{x_{i-1}})$, $x_i \in [1, k]$, $c_{x_i} \in c$, and the chunk based probability values $P(C_i = c_{x_i} \mid C_{i-1} = c_{x_{i-1}})$, $i \in [1, n]$, provide a concept transfer map to summarize the transfer probabilities between concepts at both sequence and chunk levels, with the temporal information inherently integrated into the final outcome.

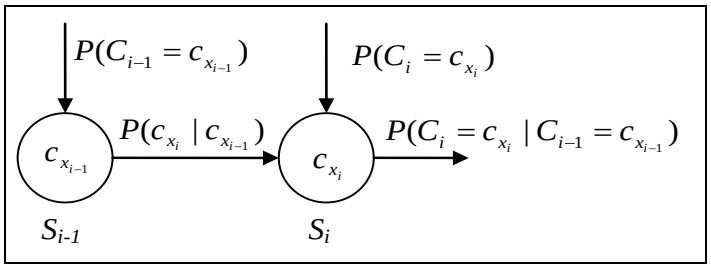


Figure 9: The graphical model of the conditional probability transferring between two consecutive chunks S_{i-1} and S_i.

6. Time Complexity Analysis

In this section, we decompose the system into two major components: vague one-class learning and one-class concept summarization, and study their time complexity in details. For ease of understanding, we denote # as the number of chunks in the data stream, where each chunk contains N instances and each instance has m attributes.

6.1 VOCL Time Complexity

The vague learning process is triggered as long as new instances form a data chunk for processing. As shown in the main procedure in Figure 5, VOCL includes three major steps including instance (local and global) weighting, weighted one-class classifier training, and classifier weighting. For instance weighting, one needs to use cross-validation to calculate each instance's local weight, and use ensemble E to determine each instance's global weight value. Assume the complexity of training a one-class classifier is $O(g(N))$, where $g(N)$ is a function with respect to the chunk size N. The local weighting involves the training of f local one-class classifiers and predictions on N instances, so the time complexity is given in Eq. (16)

$$T_{local}=O\left(f\cdot g\left(\frac{f-1}{f}N\right)\right)+O(N) \tag{16}$$

For global weighting, one needs to use ensemble (which contains k classifiers) to predict all instances in the most recent chunk, which is equivalent to the time complexity given in Eq. (17).

$$T_{global}=O(k\cdot N) \tag{17}$$

Combining Eqs. (16) and (17), the time complexity of the instance weighting process is given in Eq. (18).

$$T_{iWeighting}=O\left(f\cdot g\left(\frac{f-1}{f}N\right)\right)+O(k\cdot N)=O(f\cdot g(N))+O(k\cdot N) \tag{18}$$

For classifier weighting, the weight calculation is a pair-wise classification process across all ensemble members, which requires a total complexity of $O(k\cdot N)$. As a result, the total time complexity of VOCL over the data stream with # chunks is still bounded by the instance weighting process.

$$T_{VOCL}=O(\#\cdot f\cdot g(N)+\#\cdot k\cdot N) \tag{19}$$

6.2 OCCS Time Complexity

For concept summarization, the time complexity includes three major parts (1) set feature extraction, (2) concept clustering, and (3) concept transfer map construction. The set feature extraction process only requires one scanning of the instances (including attribute values) to build virtual samples, so the time complexity for this step is $O(m\cdot N)$ for each chunk. The time complexity of the concept clustering is dependent on the complexity of the clustering procedure. In our experiments, we used k-means clustering, so the complexity is $O(\#\cdot l\cdot m)$, where l denotes the number of iterations in the clustering process. For concept transfer map construction, we need to recursively solve Eq. (14), based on the Maximum Likelihood principle, to find optimal clustering solutions. Assume the number of clusters (concepts) specified by users is c, the above process requires, in the worst scenario, $1+2+\ldots+c=O(c^2)$ clustering process, which results in $O(\#\cdot l\cdot c^2\cdot m)$ time complexity in total. As a result, the time complexity for the concept summarization process is given in Eq.(20)

$$T_{OCCS}=O(\#\cdot m\cdot N)+O(\#\cdot l\cdot m)+O(\#\cdot l\cdot c^2\cdot m)=O(\#\cdot m\cdot N)+O(\#\cdot l\cdot c^2\cdot m) \tag{20}$$

6.3 System Time Complexity

Combing Eqs. (19) and (20), the total system time complexity is given in Eq. (21), where the first term is dominated by g(), which is the time complexity for training a one-class classifier, and the second term is dominated by either N or $l\cdot c^2$.

$$T_{OCLS}=O(\#\cdot f\cdot g(N)+\#\cdot k\cdot N)+O(\#\cdot m[N+l\cdot c^2]) \tag{21}$$

In practical settings, the values of f, k, m, l, and c are relatively small, in comparison with the chunk size N, we can therefore simplify them as a small variable $\delta<<N$, and the product between l and c^2 can be simplified as $l\cdot c^2\approx N$. As a result, the time complexity in Eq. (21) can be simplified as shown in Eq. (22),

$$T_{OCLS}=O(\#\cdot\delta\cdot g(N))+O(\#\cdot\delta\cdot N) \tag{22}$$

Depending on the actual time complexity for training a one-class classifier g(·), the total time complexity in Eq.(22) can be either dominated by the first term if g(·) is of higher order than linear or dominated by the second term if g(·) is of lower order than linear. In practice, most learning algorithms (including one-class learning) are of higher order than linear (*e.g.*, the typical time complexity for training a SVM classifier is $O(N^2)$), we assert that *the time complexity of the whole system is bounded by the complexity of the underlying one-class learner*. Assume it takes x amount of time for training a one-class classifier from a data chunk, the whole system complexity is linear with respect to both the x and the total number of chunks in the stream.

7. Experiments

7.1 Experimental Settings

We implement the proposed VOCL and OCCS algorithms using Java platform and WEKA data mining tools [42]. The one-class classifiers used in the experiments are trained using the one-class SVM provided in the LibSVM [43] package.

Benchmark Methods & Parameters: For comparison purposes, we implement three benchmark methods, LOCL, EOCL, and FOCL, which have been introduced in Section 2.2. The proposed vague one-class learning module is denoted by VOCL. For all methods, we use one-class SVM provided in the LibSVM as the generic one-class learner (using default parameter settings and Gaussian kernel). To make fair comparisons, both EOCL and VOCL use the same number of classifiers (k) to form the ensembles. To save space, we omit the details of the algorithm performance with respect to the chunk size (N), ensemble size (k) *etc*. because the impact of these factors has been addressed more or less in existing stream data mining literature [10-14]. Instead, we use chunk size N=1000, ensemble size k=10, local filter folds f=10, and weighted sampling ensemble size π=10 (as defined by Eq.(9)), for all streams. The purpose of fixing these parameters is to fully investigate and compare algorithm performance under different vague learning scenarios.

Measures: The majority experimental comparisons are based on the prediction accuracies on chunk S_{i+1}, assuming chunks …S_{i-2}, S_{i-1}, S_i have been observed so far. Because we are mainly interested in algorithms' performance across the whole stream, unless specified otherwise, we normally report the average accuracy (and the standard deviation) of each method over all data chunks.

Positive Class & User Interest Models: Because one-class learning only needs one class of samples (*i.e.*, positive samples) for training, whereas our benchmark data streams [44] contain more than one classes. To provide positive samples for one-class learning, we use the following four user interest models to select one particular class as the positive class, and treat samples from the rest of classes as non-positive samples.

To simulate user interest changes, we employ the following four models: constant, regular shifting, probability shifting, and confined probability shifting, where the first three are used for the VOCL concept learning and confined probability shifting is used for the OCCS concept summarization. In the constant model, users are interested in one particular class over the whole data stream. In the regular shifting model, users regularly shift theirs interest, from one class to another class, after a fixed number of chunks. In probability shifting model, users change their interests with the probability given in Eq. (23), where *Elaps* denotes the number of chunks the currently selected positive class has elapsed so far. For majority experiments, we use the probability shifting model, because it's closer to real-world models. The confined probability shifting is mainly used to evaluate whether the OCCS concept summarization module can indeed recapture the concept transfer relationship from the data, assume that each pair of concepts follow some predefined state transfer probability values (please refer to Section 6.4.2 for detailed explanations).

$$p(t) = \frac{1}{1 + e^{(-0.5 \times Elaps)}} \tag{23}$$

Vague Labeling: To provide vague labels for each data chunk, we employ a random vague labeling and a clustering vague labeling approach in our experiments. In *random vague labeling*, users are allowed to specify the percentage of labeled instances (α) and the percentage of genuine positive samples (β) in the labeled subset of each chunk. For each chunk S_i, instances are randomly selected to be included in the positive sample set (PS_i) to satisfy the constraints (α and β) provided by users. In some circumstances, α and β may not be satisfied simultaneously. For such a case, the labeling will first try to build a label set with the required size (α) and then try to ensure that the genuine positive samples in the labeled set satisfying the β constraint. For *clustering vague labeling*, users first apply simple k-means clustering to each data chunk and merges samples into a number of clusters. After that, clusters are sorted based on the purity with respect to the selected

positive class (*i.e.*, the percentage of genuine positive samples in each cluster) in a descending order. The clusters are selected from the top to the bottom of the list, with instances in the selected clusters included the labeled set, until the size of the labeled set reaches the α constraint. In clustering vague labeling, users cannot control the percentage of genuine positive samples in the labeled set, because a cluster can have any percentage of genuine positive samples. In the experiments, the purities of the data chunks are also reported if necessary.

Data Streams: Four benchmark streams downloaded from the Stream Data Mining Repository [44] are used in our experiments.

Sensor data stream contains information (temperature, humidity, light, and sensor voltage) collected from 56 sensors deployed in Intel Berkeley Research Lab, as shown in Figure 10. The whole stream contains consecutive information recorded over a 2 months period (1 reading per 1-3 minutes). We select sensors from four regions (the ellipses), and the learning task is to correctly identify which region a particular reading is coming from. The processed stream has four classes (*i.e.*, four regions) and 1051229 samples, each of which has five dimensions.

Power contains hourly power supply of an electricity company from two sources: power supply from the main grid and power transformed from other grids. The learning task of this stream is to predict which hours (one out of the 12 periods from (0,2], (2,4],…, (22,24]) the current power supply belongs to. The whole stream contains 29,928 samples, each of which has four dimensions.

KDD-99 was collected from the KDD CUP challenge in 1999, and the task is to build predictive models capable of distinguishing between intrusions and normal connections. The original data (10% sampling) contain 41 features, 494,021 samples, and over 22 intrusion types. We select three major classes (Normal, Neptune, and Smurf) to form a data stream with 485,269 instances.

HyperP is a synthetic data stream containing gradually evolving (drifting) concepts defined by Eq. (24), where the value a_j, j=1, 2,.., d, controls the shape of the decision surfaces, and the value $f(x)$ determines the class label of each instance x. The concept drifting of the data streams is simulated and controlled through the following parameters [24, 29]: (1) t, controlling the magnitude of the concept drifting; (2) p, controlling the number of attributes whose weights are involved in the change; and (3) h and $g \in \{-1, 1\}$, controlling the weight adjustment direction for attributes involved in the change. After the generation of each instance x, a_i is adjusted continuously by $g \cdot t$ / M (as long as a_i is involved in the concept drifting). Meanwhile, after the generation of M instances, there is an h percentage of chances that weight change will inverse its direction, *i.e.*, $g = -g$ for all attributes a_i involved in the change. In our experiments, the HyperP stream has five classes and 100,000 instances, each of which contains d=10 dimensions. The concept drifting involves p=5 attributes, and attribute weights change with a magnitude of t=0.1 in every M=2000 instances and weight adjustment inverses the direction with h=20% of chance.

$$f(x) = \sum_{j=1}^{d-1} a_j \cdot \frac{(x_j + x_{j+1})}{x_j} \tag{24}$$

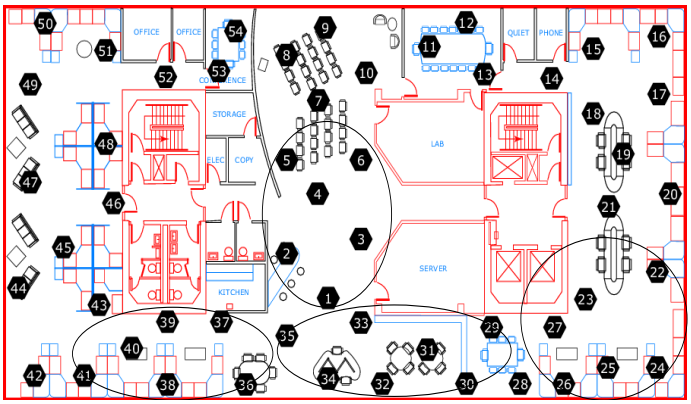


Figure 10: the sensor distribution map deployed in Intel Berkeley Research Lab. Each number denotes a sensor location, and the large ellipses denote the sensor regions used in this paper (revised from http://db.csail.mit.edu/labdata/labdata.html).

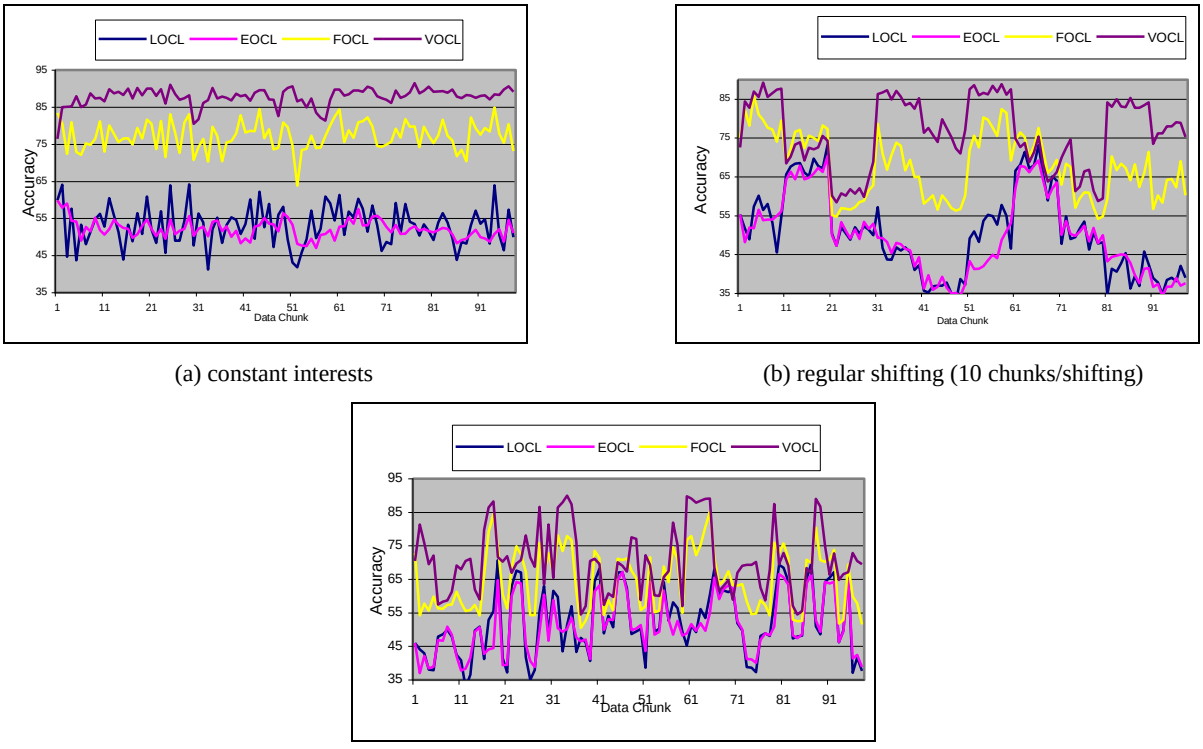
7.2 VOCL Concept Learning Results

7.2.1 Random-Based Vague Labeling Comparisons

Random vague labeling allows users to control the size of the positive sample set (α) and its purity levels (β), so we can investigate detailed algorithm performances under fully controlled vague labeling scenarios.

In Figure 11, we compare all four learning algorithms (LOCL, EOCL, FOCL, and VOCL) under three user interest models: constant, regular shifting, and probability shifting. The results from the constant user interest model (Figure 9(a)) clearly show that four methods are roughly separated into three tiers with VOCL outperforming FOCL, and FOCL further outperforming LOCL and EOCL. In fact, such ranking is also valid for all user interest models in Figure 11. Although we do expect that LOCL and EOCL to be ineffective for vague one-class learning, the results in Figure 11 actually show that one-class ensemble learning (EOCL) is not doing any better than local learning, regardless of whether the user interests are constant or shift back and forth. For one-class streams with vaguely labeled samples, simply aggregating classifiers across data chunks to form an ensemble without differentiating sample types (genuine positive or false positive) may not bring any improvement at all. The results from FOCL further support our hypothesis and show that significant performance gain can be achieved by refining vaguely labeled samples in the current and most recent chunks.

When comparing results across three user interest models, we can observe that although VOCL can outperform FOCL, it becomes increasingly difficult for VOCL to achieve large performance gains when user interests change in a random manner (*i.e.* probability shifting). This is because when user interests remain stable, VOCL can leverage information across multiple chunks to strengthen the prediction accuracy, whereas FOCL can only utilize information from neighboring chunks for learning. On the other hand, as users shift their interests away from the current class, the majority models trained from historical data chunks should be discarded. FOCL naturally adapts to such environments by using only two data chucks for training, so a shifting interest can be quickly adjusted. For VOCL, it will take at least *k*-1 chunks before the old concepts can be completely discarded. In short, although FOCL employs a filtering mechanism to leverage information from neighboring chunks, it suffers from the deficiency/risk of direct instance exclusion (which may eliminate genuine positive samples) and inclusion (which may include false-positive samples), and insufficient utilization of global knowledge from the historical data.

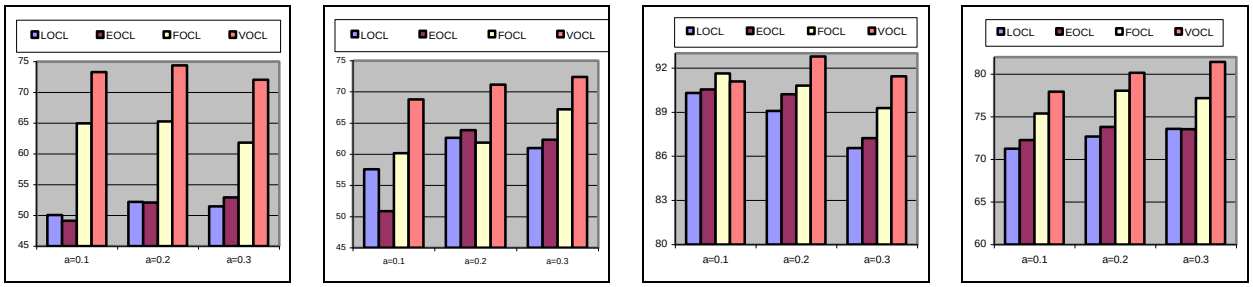


(a) constant interests

(b) regular shifting (10 chunks/shifting)

(c) probability shifting

Figure 11: Chunk-by-chunk accuracy comparisons with respect to different user interest models (HyperP data stream). Positive samples in each chunk are labeled using random vague labeling with α=0.2 and β=0.5.



(a) HyperP (b) KDD-99 (c) Power (d) Sensor

Figure 12: Average prediction accuracies with respect to different percentages of labeled samples (α) in each chunk (the *x*-axis denotes α values, and β is fixed to 0.5 for all streams, using probability shifting user interest model). For each α value, the bars from left to right correspond to LOCL, EOCL, FOCL, and VOCL.

In Figure 12, we further compare different methods with varying sizes of labeled set (α). In short, the results largely support our conclusion that when positive samples are (randomly) vaguely labeled, VOCL provides effective solution to differentiate vaguely labeled positive samples and leverage information across multiple chunks for learning.

7.2.2 Clustering-Based Vague Labeling Comparisons

In this subsection, we compare algorithm performance under scenarios that users can merge instances into clusters for labeling. In our experiments, we first use *k*-means to merge instances in S_i into *k* clusters, and then let users label one or multiple clusters as positive until the labeled set reaches size α (please refer to Section 6.1 for detailed procedures). In Figure 13, we first report the average accuracies with respect to different *k* values for *k*-means clustering (using the probability shifting model).

As the number of clusters (*i.e.*, the *k* value) grows from 3 to 50, we can observe that the purities of the labeled sample sets continuously improve. This is easy to understand because larger cluster numbers imply smaller size clusters, so the labeling process may have a better chance to select clusters containing more genuine positive samples. Interestingly, the results in Figure 11 show that while the purity of the labeled set continuously improves, no significant improvement is actually observed for all methods, except VOCL which only receives a small amount of performance gains. We believe that this is mainly because genuine positive samples selected by random vague labeling are usually more representative [45], from a supervised learning perspective, than those selected by clustering vague labeling. Indeed, in random vague labeling, genuine positive samples are randomly picked which may represent the underlying learning problem in a best way. For clustering vague labeling, genuine positive samples are selected by clusters. Because instances within one cluster usually represent a specific aspect of a big problem, genuine positive samples selected from this approach intend to be less representative. An extreme example is that if we have 100 identical instances in a chunk, the clustering approach will merge all 100 instances into one cluster. But selecting all instances in this cluster for training is obviously not a good idea (if the number of instances selected for training is limited) because the 100 instances are, in fact, one single instance and are not representative for training at all.

In Table 1, we report the algorithm performances by specifying different sizes of positive set (α) with a fixed cluster number (k=20) for all chunks. The results confirm that as the size of the labeling set grows, it is mostly helpful for all methods to receive performance gain.

The main observations from this section are threefold: (1) the VOCL learning framework can outperform FOCL and other methods for one-class data streams labeled by clustering based batch-labeling; (2) VOCL is much less sensitive than other methods w.r.t. the purities and the representativeness of the samples. In other words, VOCL can not only tolerate vaguely labeled samples, it can also work effectively on a training set where genuine positive samples are relatively less effective to represent the underlying learning concepts; and (3) comparing random vague labeling and clustering vague labeling, the former has a better capability of building a labeling set preserving the original learning concepts, whereas the latter is more effective in helping users identify positive samples.

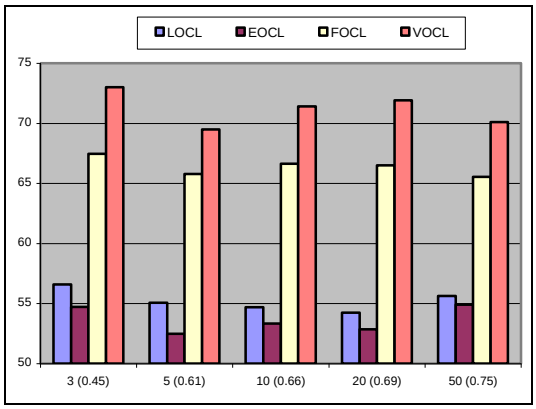


Figure 13: The average prediction accuracies with respect to different cluster numbers in vague clustering labeling (k-means clustering), the *x*-axis denotes the cluster number (the value in the bracket following each number shows the actual purity of the labeled samples), and the *y*-axis denotes the prediction accuracies of all five methods.

Table 1: Average prediction accuracies w.r.t. different percentages of labeled samples (α) in each chunk (using the probability shifting model for user interests)

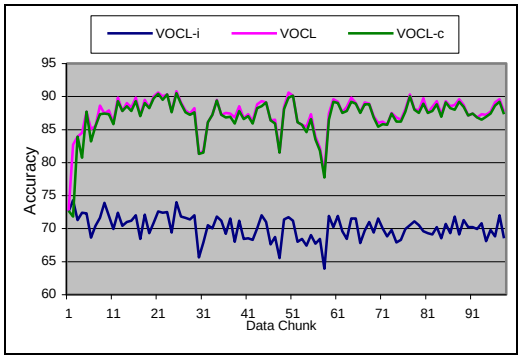
	α	*Purity*	LOCL	EOCL	FOCL	VOCL
HyperP	0.1	0.70	$56.22_{\pm 14.20}$	$56.40_{\pm 11.49}$	$67.67_{\pm 10.47}$	**$71.95_{\pm 9.89}$**
	0.2	0.69	$54.26_{\pm 10.24}$	$52.84_{\pm 10.30}$	$66.49_{\pm 11.03}$	**$71.92_{\pm 9.05}$**
	0.3	0.57	$51.70_{\pm 11.58}$	$49.35_{\pm 7.15}$	$65.58_{\pm 10.09}$	**$71.05_{\pm 9.91}$**
	0.4	0.53	$50.45_{\pm 11.58}$	$49.03_{\pm 9.73}$	$64.78_{\pm 11.32}$	**$70.57_{\pm 9.35}$**
KDD-99	0.1	0.97	$70.48_{\pm 12.23}$	$69.97_{\pm 9.78}$	$74.12_{\pm 12.88}$	**$80.35_{\pm 8.33}$**
	0.2	0.95	$71.60_{\pm 14.49}$	$70.84_{\pm 8.64}$	$74.56_{\pm 12.04}$	**$81.60_{\pm 9.58}$**
	0.3	0.92	$68.85_{\pm 13.49}$	$69.92_{\pm 10.12}$	$75.35_{\pm 11.34}$	**$80.23_{\pm 8.83}$**
	0.4	0.84	$65.47_{\pm 14.22}$	$67.44_{\pm 9.80}$	$74.48_{\pm 12.55}$	**$80.51_{\pm 9.24}$**
Power	0.1	0.78	$90.22_{\pm 1.96}$	**$91.53_{\pm 0.98}$**	$91.48_{\pm 0.93}$	**$91.00_{\pm 0.71}$**
	0.2	0.41	$87.81_{\pm 1.39}$	$90.33_{\pm 0.86}$	**$91.25_{\pm 0.97}$**	**$91.17_{\pm 0.83}$**
	0.3	0.29	$85.12_{\pm 1.52}$	$84.33_{\pm 0.72}$	$86.49_{\pm 0.89}$	**$89.38_{\pm 0.92}$**
	0.4	0.21	$82.81_{\pm 1.78}$	$79.46_{\pm 0.81}$	$85.93_{\pm 0.92}$	**$89.07_{\pm 0.67}$**
Sensor	0.1	1.00	$72.44_{\pm 2.27}$	$71.57_{\pm 1.87}$	$74.51_{\pm 2.03}$	**$77.27_{\pm 1.24}$**
	0.2	0.97	$74.22_{\pm 3.02}$	$74.29_{\pm 1.45}$	$76.54_{\pm 2.77}$	**$79.72_{\pm 1.76}$**
	0.3	0.78	$76.89_{\pm 2.55}$	$77.69_{\pm 1.92}$	$78.09_{\pm 2.59}$	**$80.80_{\pm 1.59}$**
	0.4	0.47	$77.02_{\pm 2.87}$	$76.92_{\pm 1.68}$	$78.97_{\pm 2.62}$	**$81.82_{\pm 1.68}$**

7.2.3 Local and Global Weighting Comparisons

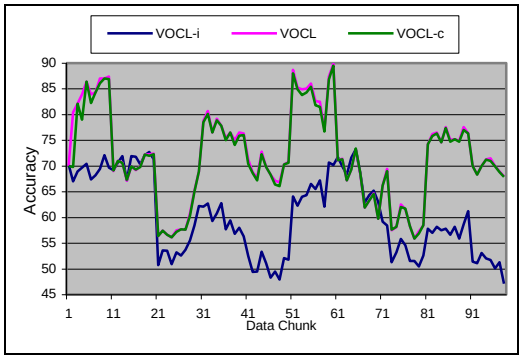
The VOCL design as shown in Figure 5 involves a two-layer weighting mechanism, at both instance and classifier levels. The coupling of the two-layer weighting mechanism raises a necessary concern on which part of the weighting (instance or classifier level) is responsible for the performance gain/loss of VOCL. To comparatively study instance and classifier weighting, as described in Sections 4.2 and 4.3, we design the following experiment to compare VOCL with its two modified versions. In our experiment, we first remove the instance weighting module from the VOCL (so all instances in each chunk have the same weight values) and keep the rest of the VOCL design as the same as shown in Figure 5. We denote this method as VOCL-i (*i.e.*, VOCL minus instance weighting). Similarly, we also remove the classifier weighting module from the VOCL (so all classifiers have the equal weight values) and keep the rest of the VOCL design unchanged, and denote this method as VOCL-c (i.e., VOCL minus classifier weighting). We apply three methods VOCL-i, VOCL-c, and VOCL to the same testbed and report their performances in Figure 14.

The results in Figure 14 clearly show that instance weighting, overall, play a major role for VOCL. For all three user interest models, as shown in Figures 14(a), 14(b), and 14(c), the prediction accuracies will drop significantly as long as the instance weighting module is removed from VOCL. On the other hand, classifier weighting, in general, has a very

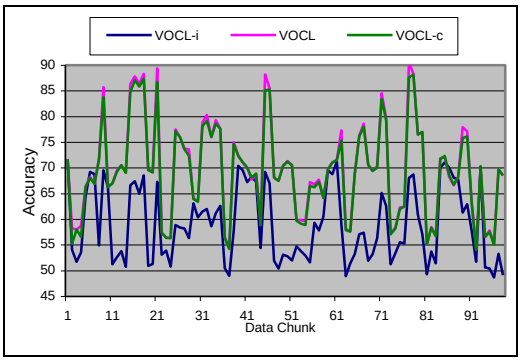
limited impact on the system performance unless the user interest changes frequently and rapidly (as shown in Figure 14(c)). If the user interest maintains stable or switches regularly (Figures 14(a) and (b)), we can observe that the deterioration of the prediction accuracy, in each chunk, is between 0.5% to 2% or so, which is far less significant than the loss due to the removing of the instance weighting module. Indeed, when samples in each chunk are vaguely labeled, instance weighting provides an effective way to differentiate genuine positive samples so the classifier trained from each chunk can have a good performance. This observation also suggests that in data stream environment, focusing on improving each single data chunk to produce good predictors at chunk level may be more beneficial than simply sticking to the high level combination of all predictors, although the latter has a clear advantage of tackling the concept drifting challenge in data streams.



(a) constant interests



(b) regular shifting (10 chunks/shifting)



(c) probability shifting

Figure 14: The comparisons of the local and global weighting mechanisms with respect to the VOCL which combines both local and global weighting. VOCL-i denotes the VOCL framework without instance weighting (Section 4.2) whereas the classifier weighting remains the same as VOCL. VOCL-c represents the VOCL framework without classifier weighting (Section 4.3) whereas the instance weighting remains the same as VOCL. The comparisons across VOCL-i, VOCL-c, and VOCL clearly show that instance weighting plays a major role for VOCL **(**HyperP data stream, positive samples in each chunk are labeled using random vague labeling with α=0.2 and β=0.5)

7.3 Concept Summarization Results

The performance of the OCCS module for concept summarization is evaluated based on the algorithm performance with respect to the clustering accuracy and concept transfer accuracy estimated from the HyperP and Sensor streams.

7.3.1 Concept Clustering Accuracy

Since the main purpose of the concept clustering is to merge chunks with similar concepts into groups, according to the samples labeled in each chunk by users, we evaluate the concept clustering performance by assessing the quality of the clustering results. More specifically, when labeling a chunk using a vague labeling approach, we assume that we know the actual concept users intended to label the chunk (in the experiments, the concept in each chunk is determined by the class

the users use for labeling). This is to say that for each instance in the virtual set, we know its genuine class label, and the assessment is made by evaluating whether the clustering approach is able to merge instances with the same class label into a cluster. For this purpose, we apply K-means clustering [46] to the virtual set with the number of clusters set as the genuine class number in the stream. For each cluster, we find the concept (*i.e.*, class) which dominants the cluster and regard that this cluster represents the class. The clustering accuracy for a cluster is the percentage of samples belonging to the dominant class, comparing to the total number of instances in the cluster. The concept clustering accuracy is the average clustering accuracy over all clusters.

In Figure 15, we report the concept clustering accuracy, with respect to different numbers of bins used to discretize each feature (under probability concept shifting model). For comparison purposes, we also compare, for each stream, the clustering accuracy of using labeled samples in each chunk only (without any weighting), so the benefits of the proposed instance weighting approach can be clearly demonstrated.

As shown in Figure 15, for HyperP stream, the clustering accuracy can reach about 50% when each feature is equally separated into 5 bins to build the feature histogram. We note that HyperP has 5 concepts, so this accuracy, although still have much room for improving, is much better than a random guess whose accuracy would be 20% only. In addition, the proposed sample weighting approach has shown to improve the accuracy by 1-5%, comparing to the approach using labeled samples only. For Sensor stream, the results do not show that sampling weighting or bin numbers have any significant impact to the clustering accuracy. This is mainly because that the feature values of the Sensor stream are relatively sparse, and the proposed discretization approach may need to be carefully tuned for each individual feature to build good set features for clustering.

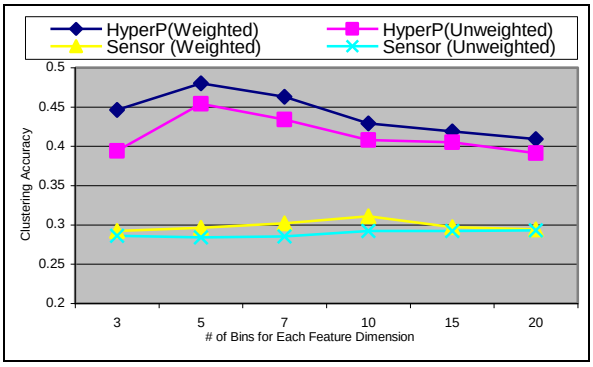


Figure 15: Concept clustering accuracy with respect to the number of bins used to discretize each feature (50-times K-means clustering average accuracy, using probability concept shifting model with α=0.1 and β=0.5)

7.3.2 Markov Model Based Concept Summarization Results

One major objective of the concept summarization is to build a concept transfer map, using Markov model, to visualize the temporal correlations of the concepts in the stream. To evaluate the algorithm performance, we employ a *confined probability shifting model* which is actually a Markov model with known state transfer probabilities. When labeling the data, we assume that users employ the given concept transfer model to label each chunk and change their labeling interests. The evaluation of the algorithm performance is to determine that to which extent the proposed method can restore the original concept transfer model behind users' labeling, by observing chunks labeled by users.

In our experiments, we assume that users are following the concept transfer model in Table 2 to label each chunk. In the model given in Table 2, we intentionally separate concepts into two major groups, with $\{c_1, c_2\}$ and $\{c_3, c_4, c_5\}$ each belonging to one group. The purpose of employing such a unique generative model is to assess whether the proposed approach is able to recapture the high-level relationship among the concepts underneath the data.

For each data stream labeled by using the generative model (G) given in Table 2, we employ the proposed concept summarization approach to rebuild a concept transfer matrix (*i.e.*, a Markov model), denoted by G', from the data. Because we don't actually know the mapping relationship between a concept in G and the concept in G', we randomly permutate each pair of rows (i,j) and each pairs of columns (i,j), until we find a permutation G'' of G' which has the smallest distance with G. The concepts in G'' are then regarded as the matching concepts with G, and their state transfer values are reported in Tables 3.

Table 2: The state transfer matrix used for concept labeling (using confined probability shifting model). Each table field (*i*, *j*) denotes the probability of the concept in the row *i* to be transferred to the concept in the column *j* (HyperP Stream with random vague labeling)

	c_1	c_2	c_3	c_4	c_5
c_1	0.5	0.4	0.1	0	0
c_2	0.4	0.3	0.2	0.1	0
c_3	0	0.1	0.3	0.4	0.2
c_4	0	0.1	0.3	0.2	0.4
c_5	0	0	0.3	0.3	0.4

Table 3: The state transfer matrix restored from the data labeled by using the generative matrix given in Table 2. Each table field (*i*, *j*) denotes the probability of the concept in the row *i* to be transferred to the concept in the column *j*. The clustering accuracy is 0.554 and the distance between this matrix and the generative matrix in Table 2.1 is about 0.023 (HyperP Stream with random vague labeling)

	c'_1	c'_2	c'_3	c'_4	c'_5
c'_1	0.214	0.286	0.214	0.143	0.143
c'_2	0.333	0.333	0.111	0.111	0.111
c'_3	0.132	0.071	0.286	0.321	0.178
c'_4	0.115	0.077	0.192	0.346	0.269
c'_5	0.048	0.0	0.381	0.286	0.286

The results in Table 3 indicate that the proposed concept summarization approach is able to largely restore the concept transfer relationships from the data. When comparing two matrices in Tables 2 and 3, we can find that although the actual values are subject to large differences (the normalized Euclidean distance between two matrices is 0.023), the overall results clearly show that concepts c'_1 and c'_2 form one group, and c'_3, c'_4, and c'_5 form the second group, which retain the same relationship as the generative model given in Table 2.

In summary, the above results assert that the proposed concept summarization approach is able to recapture concept transfer relationships from the data and help users summarize concepts labeled by them over the whole stream.

8. Related Work

The proposed work is closely related to existing research on one-class learning and stream data mining.

One-class learning [1-14] refers to a special type of supervised learning task where only one class of labeled samples is available for training. Three types of one-class learning methods commonly exist, and the most popular approach is to regard the one-class learning as an optimization problem where the objective is to either find a hyper-ball to enclose all positive samples [6] or to find a hyper-plane [7] to separate all positive samples from the origin. The second type of one-class learning approach is to learn an autoassociation neural network [14] which intends to reproduce the input signal such that the network can work as a recognition machine to recognize positive signals. For applications containing both labeled (positive) and unlabeled samples, a third type of one-class learning solution is to transform the learning as some binary classification problems [8-10]. In comparison, although vague one-class learning is closely related to the one-class learning, our focus is to consider vaguely labeled samples, where most, if not all, one-class leaning methods are inadequate to address the problem.

Stream data mining [19-33] traditionally concerns the problem of discovering patterns or building prediction models from continuous volume stream data. Two major challenges in stream data mining include (1) continuously increasing data volumes, and (2) the drifting of the decision concepts. These two challenges are traditionally solved by using either an incremental learning [21-22] or an ensemble learning approach [23-24]. For incremental learning, the problem is to build a model from a small portion of the stream data, and continuously update the model by using newly arrived samples. For ensemble learning, a number of base classifiers are built from different portions of stream data, and the final goal is to combine the models to form an ensemble classifier for prediction. Many algorithms exist for using an ensembling based method for stream-oriented applications, such as active learning [47], learning with imbalanced sample distributions [31], proactive mining [32], and learning with labeled and unlabeled samples [33]. For all methods in this category, the samples are assumed to be accurately labeled (except a recent stream data cleansing method [29], whose objective is to cleanse

noisy samples in stream data), whereas in the proposed research we intend to address vaguely (or weakly) labeled samples in data streams.

The problem of employing one-class learning for data streams was recently addressed by Li et al. [8] in their positive unlabeled learning method, which refines positive samples and includes samples from the most recent data chunk (the one proceeding to the current chunk) for data stream classification. Our work differs from Li et al's work on two aspects. First, we employ ensemble learning and instance weighting to handle vaguely labeled samples, whereas Li's method uses cross-chunk sample inclusion or exclusion to refine the data chunks for learning. Second, Li's method does not address the concept summarization problem whereas our framework does. In a conference version of our recent work [28], we addressed the vague one-class learning for data streams, but the problem of concept summarization was not addressed in our previous efforts or by any other research endeavors.

9. Conclusions and Remarks

In this paper, we formulated a new research problem of vague one-class learning and concept summarization for data streams. We argued that in data stream environments, providing fast and accurate labeling information is crucial but difficult to realize, mainly because that existing instance-based labeling approaches are expensive and time consuming. On the other hand, while all labeled samples in a one-class stream are marked as positive, the change of the user labeling interests makes it necessary to recapture and summarize users' labeling concepts over the whole stream.

Following the above motivations, we advocated, in this paper, a vague labeling paradigm which allows users to merge instances into groups and label positive groups instead. The vague labeling approach, nevertheless, raises three special challenges. First, a vaguely labeled training set may contain non-positive instances. Second, users may shift their interests at any time and the concepts underneath the data may also change gradually. Last, as data volumes continuously grow, it makes traditional one-class learning algorithms incapable of handling stream data. To solve these problems, we proposed a One-Class Learning and Summarization (OCLS) system which uses a Vague One-Class Learning (VOCL) and a One-Class Concept Summarization (OCCS) model to fulfill the concept learning and summarization goal. Experimental results on four data streams confirmed that VOCL significantly outperforms its rival peers to support one-class learning for vaguely labeled data streams. The proposed OCCS is also able to rebuild the concepts and recapture their transfer relationships over the stream.

As existing tools for one-class learning are extremely limited, many problems for one-class data stream learning remain wide open. In the paper, we have shown that labeling quality plays a fundamental role for one-class learning. Proposing new solutions for users to identify important samples for labeling [30, 47] is an important direction which should be addressed in the future. In addition, the concept summarization approaches reported in the paper still have much room for improvement. Devising efficient and effective methods to help build concept summaries for one-class data streams is another direction worth of pursuing.

Acknowledgement

The authors would like to thank Dr. Xindong Wu from the University of Vermont for his contributions to the conference version [28] of this work.

References

[1]. R. Perdisci, G. Gu, & W. Lee, Using an Ensemble of One-Class SVM Classifiers to Harden Payload-based Anomaly Detection Systems, *Proc. of ICDM*, 2006.

[2]. L. Manevitz & M. Yousef, One-Class SVMs for Document Classification, *Journal of Machine Learning Research*, 2:139-154, 2002.

[3]. K. Goh, E. Chang, & B. Li, Using One-Class and Two-Class SVMs for Multiclass Image Annotation, *IEEE Transactions on Knowledge and Data Engineering*, 17(10):1333-1346, Oct. 2005.

[4]. M. Koppel & J. Schler, Authorship Verification as a One-Class Classification Problem, *Proc. of ICML*, 2004.

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

[6]. D. M. J. Tax. *One-Class Classification, Concept Learning in the Absence of Counter Examples*. PhD Thesis, Delft University of Technology, Delft, Netherland, 2001.

[7]. B. Sch¨olkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, & RC Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13:1443–1471, 2001

[8]. X. Li, P. Yu, B. Liu, & S. Ng, Positive Unlabeled Learning for Data Stream Classification, *in Proc. of SDM*, 2009.

[9]. C. Elkan & K. Noto, Learning Classifiers from Only Positive and Unlabeled Data, *in Proc. of SIGKDD*, 2008.

[10]. B. Liu, Y. Dai, W. S. Lee, P. S. Yu, and X. Li, Building Text Classifiers Using Positive and Unlabeled Examples, *in Proc. of ICDM*, 2003.

[11]. M. Yousef, S. Jung, L. Showe, & M. Showe, Learning from Positive Examples when the Negative Class is Undetermined-microRNA gene identification, *Algorithms for Molecular Biology*, 3:2, 2008.

[12]. Y. Chen, X. Zhou, & T. Huang, One-Class SVM for Learning in Image Retrieval, in *Prof. of International Conference on Image Processing*, 2001.

[13]. Q. Wang & L. Lopes, One-Class Learning for Human-Robot Interaction, in *Emerging Solutions for Future Manufacturing Systems*, pp.489-498, Springer, 2005.

[14]. N. Japkowicz, *Concept-Learning in the Absence of Counter-Examples: An Autoassociation-Based Approach to Classification*, Ph, D Dissertation, Rutgers, the State University of New Jersey, 1999.

[15]. L. Breiman, J. Friedman, C. Stone, & R. Olshen, *Classification and Regression Trees*, Chapman & Hall /CRC, 1984.

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

[17]. X. Zhu, X. Wu, & Q. Chen, Eliminating Class Noise in Large Datasets, *Proc. of ICML*, 2003.

[18]. X. Zhu & X. Wu, Class Noise vs. Attribute Noise: A Quantitative Study of Their Impacts, *Artificial Intelligence Review*, 22:177-210, 2004.

[19]. C. Aggarwal, *Data Streams: Models and Algorithms*, Springer, 2007

[20]. I. Akyildiz, W. Su, Y. Sankarasubramaniam, & E. Cayirci, Wireless Sensor Networks: A Survey, *Computer Networks*, 38:393-422, 2002.

[21]. P. Domingos & G. Hulten, Mining high-speed data streams, *Proc. of KDD*, 2000.

[22]. G. Hulten, L. Spencer, & P. Domingos, Mining time-changing data streams, *Proc. of KDD*, 2001.

[23]. W. Street & Y. Kim, A streaming ensemble algorithm (SEA) for large-scale classification, *Proc. of KDD*, 2001.

[24]. H. Wang, W. Fan, P. Yu, & J. Han, Mining concept-drifting data streams using ensemble classifiers, *Proc. of KDD*, 2003

[25]. H. Wang *et al.*, Suppressing model overfitting in mining concept-drifting data streams, *Proc. of KDD*, 2006.

[26]. G. Widmer and M. Kubat, Learning in the presence of concept drift and hidden contexts, *Machine Learning*, vol. 23, pp. 69-101, 1996.

[27]. P. Zhang, X. Zhu, and Y. Shi, Categorizing and Mining Concept Drifting Data Streams, in *Proc. of the 14th KDD Conference*, 2008.

[28]. X. Zhu, X. Wu, and C. Zhang, One-Class Vague Learning for Data Streams, in *Proc. of the 9th IEEE International Conference on Data Mining*, Miami, FL, 2009.

[29]. X. Zhu, P. Zhang, X. Wu, D. He, C. Zhang, and Y. Shi, Cleansing Noisy Data Streams, *Proc. Of the 8th IEEE International Conference on Data Mining (ICDM)*, Pisa, Italy, December, 2008.

[30]. W. Fan, Y. Huang, H. Wang, and P. Yu, Active Mining of Data Streams, in *Proc. of SIAM International Conf. on Data Mining*, 2004.

[31]. J. Gao, W. Fan, J. Han, and P. Yu, A General Framework for Mining Concept-Drifting Data Streams with Skewed Distributions, in *Proc. of SIAM International Conference on Data Mining*, 2007

[32]. Y. Yang, X. Wu, & X. Zhu, Combining proactive and reactive predictions of data streams, in *Proc. of KDD*, 2005

[33]. P. Zhang, X. Zhu, and L. Guo, Mining Data Streams with Labeled and Unlabeled Training Examples, in *Proc. of ICDM*, 2009

[34]. K. Nigam, A. Mccallum, S. Thrun, & T. Mitchell, Text Classification from Labeled and Unlabeled Documents using EM, *Machine Learning*, 1-34, 1999.

[35]. T. Dietterich, Ensemble methods in machine learning, *Proc. of the 1st Workshop on Multiple Classifier Systems*, 2000.

[36]. D. Newman, S. Hettich, C. Blake, & C Merz. *UCI Repository of machine learning*, 1998.

[37]. C. Brodley & M. Friedl, Identifying Mislabeled Training Data. *Journal of AI Research* (JAIR) 11: 131-167, 1999.

[38]. J. Von Neumann, *Various Techniques Used in Connection with Random Digits*, Nat'l Bureau of Standards Applied Math. Series 12, pp. 36-38, 1951.

[39]. B. Zadrozny, J. Langford, & N. Abe, Cost-Sensitive Learning by Cost-Proportionate Example Weighting, *Proc. of ICDM*, 2003.

[40]. S. P. Meyn and R.L. Tweedie, 2005. *Markov Chains and Stochastic Stability*. Second edition, Cambridge University Press, 2008.

[41]. A. Dempster, N. Laird, & D. Rubin, Maximum Likelihood from Incomplete Data via the EM Algorithm, *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1-38, 1977.

[42]. I. Witten & E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd Edition, Morgan Kaufmann, 2005.

[43]. C.-C. Chang & C.-J. Lin, LIBSVM : a library for support vector machines, http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[44]. X. Zhu, Stream Data Mining Repository, http://www.cse.fau.edu/~xqzhu/stream.html, 2009.

[45]. X. Zhu and X. Wu, Scalable Representative Instance Selection and Ranking, in *Proc. of the 18th International Conference on Pattern Recognition (ICPR)*, 2006.

[46]. MacQueen, J. B. (1967). Some Methods for classification and Analysis of Multivariate Observations. In *Proc. of 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

[47]. X. Zhu, P. Zhang, X. Lin, & Y. Shi, Active Learning from Data Streams, in *Proc. of the 7th IEEE ICDM Conference*, 2007.

[48]. C. Aggarwal, On classification and segmentation of massive audio data streams, *Knowledge and Information Systems*, 20(2):137-156, 2009.

[49]. X. Dang, W. Ng, K. Ong, Online mining of frequent sets in data streams with error guarantee, *Knowledge and Information Systems*, 16(2):245-258, 2008.

[50]. M. Cho, J. Pei, K. Wang, Answering ad hoc aggregate queries from data streams using prefix aggregate trees, *Knowledge and Information Systems*, 12(3):301-329, 2007.

[51]. X. Zhu, X. Wu, Y. Yang, Effective classification of noisy data streams with attribute-oriented dynamic classifier selection, *Knowledge and Information Systems*, 9(3):339-363, 2006.

[52]. B. Jiang, M. Zhang, and X. Zhang, OSCAR: One-class SVM for accurate recognition of cis-elements, *Bioinformatics*, vol. 23, no. 21, pp.2823-2828, 2007.