Catching the Trend: A Framework for Clustering Concept-Drifting Categorical Data

Hung-Leng Chen, *Member*, *IEEE*, Ming-Syan Chen, *Fellow*, *IEEE*, and Su-Chen Lin

Abstract—Sampling has been recognized as an important technique to improve the efficiency of clustering. However, with sampling applied, those points that are not sampled will not have their labels after the normal process. Although there is a straightforward approach in the numerical domain, the problem of how to allocate those unlabeled data points into proper clusters remains as a challenging issue in the categorical domain. In this paper, a mechanism named MAximal Resemblance Data Labeling (abbreviated as MARDL) is proposed to allocate each unlabeled data point into the corresponding appropriate cluster based on the novel categorical clustering representative, namely, N-Nodeset Importance Representative(abbreviated as NNIR), which represents clusters by the importance of the combinations of attribute values. MARDL has two advantages: 1) MARDL exhibits high execution efficiency, and 2) MARDL can achieve high intracluster similarity and low intercluster similarity, which are regarded as the most important properties of clusters, thus benefiting the analysis of cluster behaviors. MARDL is empirically validated on real and synthetic data sets and is shown to be significantly more efficient than prior works while attaining results of high quality.

Index Terms—Data mining, categorical clustering, data labeling.

1 Introduction

Data clustering is an important technique for exploratory data analysis and has been the focus of substantial research in several domains for decades [22], [23]. The problem of clustering is defined as follows: Given a set of data objects, the problem of clustering is to partition data objects into groups in such a way that objects in the same group are similar while objects in different groups are dissimilar according to the predefined similarity measurement. Therefore, clustering analysis can help us to *gain insight into the distribution of data* [18].

However, a difficult problem with learning in many real-world domains is that the concept of interest may depend on some *hidden context*, not given explicitly in the form of predictive features. In other words, the concepts that we try to learn from those data *drift with time* [21], [30], [31]. For example, the buying preferences of customers may change with time, depending on the current day of the week, availability of alternatives, discounting rate, etc. As the concepts behind the data evolve with time, *the underlying clusters may also change considerably with time* [1]. Performing clustering on the entire time-evolving data not only *decreases the quality of clusters* but also *disregards the expectations of users*, which usually *require recent clustering results*.

- *H.-L. Chen and S.-C. Lin are with the Department of Electrical Engineering, National Taiwan University, No. 1, Sec. 4, Roosevelt Road, Taipei 106, Taiwan, ROC. E-mail: {kidd, sclin}@AEA-arbor.ee.ntu.edu.tw.*
- *M.-S. Chen is with the Department of Electrical Engineering and the Graduate Institute of Communication Engineering, National Taiwan University, Taipei 106, Taiwan, ROC. E-mail: mschen@AEA-cc.ee.ntu.edu.tw.*

Manuscript received 13 Jan. 2008; revised 13 May 2008; accepted 12 Aug. 2008; published online 10 Sept. 2008.
Recommended for acceptance by L. Wipo.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2008-01-0022.
Digital Object Identifier no. 10.1109/TKDE.2008.192.

The problem of clustering time-evolving data in the numerical domain has been explored in the previous works [1], [5], [6], [8], [9], [12], [24], [32]. However, this problem has not been widely discussed in the *categorical domain* with the exception of [25] for Web log transactions. Actually, categorical attributes also prevalently exist in real data with drifting concepts. For example, buying records of customers, Web logs that record the browsing history of users, or Web documents often evolve with time. Previous works on clustering categorical data focus on doing clustering on the entire data set and do not take the drifting concepts into consideration. Therefore, the problem of *clustering time-evolving data in the categorical domain remains a challenging issue.*

As a result, a framework for *performing clustering on the categorical time-evolving data* is proposed in this paper. Instead of designing a specific clustering algorithm, we propose a *generalized clustering framework* that utilizes existing clustering algorithms and detects if there is a drifting concept or not in the incoming data. Fig. 1 shows our entire framework of performing clustering on the categorical time-evolving data. In order to detect the drifting concepts, the *sliding window* technique is adopted. Sliding windows conveniently eliminate the outdated records [35], and the sliding windows technique is utilized in several previous works on clustering time-evolving data in the numerical domain [1], [6], [8], [12]. Therefore, based on the sliding window technique, we can test the latest data points in the current window if the characteristics of clusters are similar to the last clustering result or not.

In [12], a similar strategy with our framework that utilizes clustering results to analyze the drifting concepts is proposed in the numerical domain. The online clustering algorithm is performed on each time frame, and several numerical characteristics such as the mean and standard deviation of cluster centers are used to represent clustering

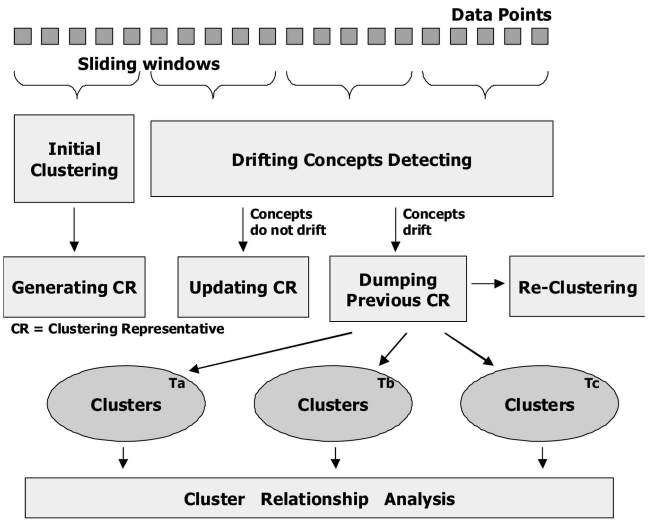


Fig. 1. The framework of performing clustering on the categorical time-evolving data.

results and detect the changes between time frames. After the change is detected, an offline voting-based classification algorithm is performed to associate each change with its correspondent event. However, in the categorical domain, the above procedure is infeasible because the numerical characteristics of clusters are difficult to define.

Therefore, for capturing the characteristics of clusters, an effective cluster representative that summarizes the clustering results is required. In this paper, a practical categorical clustering representative, named "*Node Importance Representative*" (abbreviated as **NIR**) [7], is utilized. NIR represents clusters by measuring the importance of each attribute value in the clusters. Based on NIR, we propose the "*Drifting Concept Detection*" (abbreviated as **DCD**) algorithm in this paper. In DCD, the incoming categorical data points at the present sliding window are first allocated into the corresponding proper cluster at the last clustering result, and the number of outliers that are not able to be assigned into any cluster is counted. After that, the distribution of clusters and outliers between the last clustering result and the current temporal clustering result are compared with each other. If the distribution is changed (exceeding some criteria), the concepts are said to drift. In the concept-drifting window, the data points will do reclustering, and the last clustering representative will be dumped out. On the contrary, if the concept is steady, the clustering representative (NIR) will be updated.

Moreover, the framework presented in this paper not only detects the drifting concepts in the categorical data but also explains the drifting concepts by analyzing the relationship between clustering results at different times. The analyzing algorithm is named "*Cluster Relationship Analysis*" (**CRA**). When the drifting concept is detected by DCD, the last clustering representative is dumped out. Therefore, each clustering representative that had been recorded represents a successive constant clustering result, and different dumped-out representatives describe different concepts in the data set. By analyzing the relationship between clustering results, we may capture the time-evolving trend that explains why the clustering results have changes in the data set.

Our contributions in this paper can be summarized as follows:

- A generalized framework of *performing clustering on the categorical time-evolving data* is proposed in this paper. In particular, this framework is independent of clustering algorithms, and any categorical clustering algorithm can be utilized in this framework.
- Based on the sliding window technique and the categorical clustering representative NIR, the algorithm DCD is proposed in this paper. In DCD, *the distributions between clustering results are tested to determine whether the concepts drift or not.*
- The CRA algorithm, which analyzes the relationships between clustering results generated by different concepts, is also presented in this paper. We may *capture the time-evolving trend in the data set by analyzing the evolving clustering results.*

This paper is organized as follows: In Section 2, we introduce the preliminaries and formulate the problem of this work. Section 3 presents the DCD algorithm, and the CRA algorithm is introduced in Section 4. Section 5 reports our performance study on real and synthetic data sets. In Section 6, we review several related works. The paper concludes with Section 7.

2 Preliminaries

In Section 2.1, the problem definition and several notations of this work are defined. After that, in Section 2.2, we will simply introduce the practical categorical clustering representative named NIR, which was presented in our previous work [7].

2.1 Problem Description

The problem of clustering the categorical time-evolving data is formulated as follows: Suppose that a series of categorical data points D is given, where each data point is a vector of q attribute values, i.e., $p_j = (p_j^1, p_j^2, \ldots, p_j^q)$. Let $A = \{A_1, A_2, \ldots, A_q\}$, where A_a is the ath categorical attribute, $1 \leq a \leq q$. In addition, suppose that the window size N is also given. The data set D is separated into several continuous subsets S^t, where the number of data points in each S^t is N. The superscript number t is the identification number of the sliding window and t is also called time stamp in this paper. For example, the first N data points in D are located in the first subset S^1. Based on the foregoing, the objective of the framework is to *perform clustering on the data set D and consider the drifting concepts between S^t and S^{t+1} and also analyze the relationship between different clustering results.*

For ease of presentation, several notations are defined as follows:

In our framework, several clustering results at different time stamps will be reported. Each clustering result $C^{[t_1,t_2]}$ is formed by one stable concept that persists for a period of time, i.e., the sliding windows from t_1 to t_2. The clustering results $C^{[t_1,t_2]}$ contain $k^{[t_1,t_2]}$ clusters, i.e., $C^{[t_1,t_2]} = \{c_1^{[t_1,t_2]}, c_2^{[t_1,t_2]}, \ldots, c_{k^{[t_1,t_2]}}^{[t_1,t_2]}\}$, where $c_i^{[t_1,t_2]}$, $1 \leq i \leq k^{[t_1,t_2]}$, is the ith cluster in $C^{[t_1,t_2]}$. If $t_1 = t_2 = t$, we simplify the

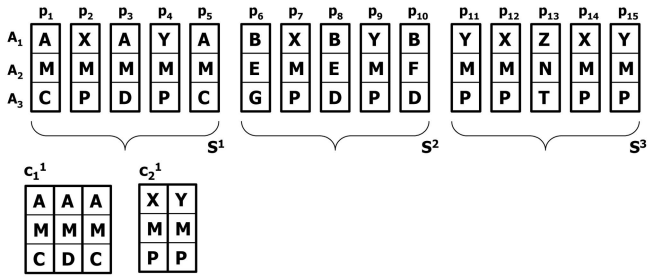


Fig. 2. An example data set where the initial clustering is performed.

superscript by t. For example, the first clustering result that is obtained from the initial clustering step is C^1. Moreover, if we do not point out a specific time stamp, the superscript will be omitted for ease of presentation. In addition, when the DCD algorithm is performed, a temporal clustering result, which is utilized to detect the drifting concept at each sliding window, will be obtained. The notation C'^t is used to represent the temporal clustering result at time stamp t. Fig. 2 shows an example of data set D with 15 data points, three attributes, and the sliding window size $N = 5$. The initial clustering is performed on the first sliding window S^1, and the clustering result C^1, which contains two clusters, c_1^1 and c_2^1, is obtained.

All of the symbols utilized in this paper are summarized in Table 1.

2.2 Node Importance Representative

The basic idea behind NIR is to represent a cluster as the distribution of the attribute values, which are called "*nodes*" in [7]. In order to measure the representability of each node in a cluster, the importance of a node is evaluated based on the following two concepts:

1. *The node is important in the cluster when the frequency of the node is high in this cluster.*
2. *The node is important in the cluster if the node appears prevalently in this cluster rather than in other clusters.*

The formal definitions of nodes and node importance are shown as follows:

Definition 1 (node). *A node, I_r, is defined as attribute name + attribute value.*

The term *node*, which is defined to represent attribute value in this paper, avoids the ambiguity that might be caused by identical attribute values. If there are two different attributes with the same attribute value, e.g., the age is in the range 50-59 and the weight is in the range 50-59, the attribute value 50-59 is confusing when we separate the attribute value from the attribute name. $Nodes$ $[age = 50\text{-}59]$ and $[weight = 50\text{-}59]$ avoid this ambiguity.

Suppose that the number of data points in cluster c_i is m_i, the node I_r that occurs in the cluster c_i is abbreviated by I_{ir}, and the frequency of the node I_{ir} is $|I_{ir}|$. In addition, the number of clusters is k. Based on the foregoing, the *node importance* is defined as follows:

TABLE 1
Summary of the Symbols Utilized in This Paper

A_a	The a-th attribute in the data set.
$C^{[t_1,t_2]}$	The clustering result from t_1 to t_2.
C^t	The clustering result on sliding window t.
C'^t	The temporal clustering result on sliding window t.
c_i	The i-th cluster in C.
$\overline{c_i}$	The node importance vector of c_i.
I_{ir}	The r-th node in c_i.
$\lvert I_{ir}\rvert$	The number of occurence of I_{ir}.
k	The number of clusters in C.
m_i	The number of data points in c_i.
N	The size of sliding window.
S^t	The sliding window t.
t	The timestamp index of sliding window.
$w(c_i, I_{ir})$	The importance of I_{ir} in c_i.
θ	The outlier threshold.
ϵ	The cluster variation threshold.
η	The cluster difference threshold.
$CM(\overline{c_i}, \overline{c_j})$	The cosine measure between cluster vectors $\overline{c_i}$ and $\overline{c_j}$.

Definition 2 (node importance). *The importance value of the node I_{ir} is calculated as the following equations:*

$$w(c_i, I_{ir}) = \frac{|I_{ir}|}{m_i} * f(I_r),$$

$$f(I_r) = 1 - \frac{-1}{\log k} * \sum_{y=1}^{k} p(I_{yr}) \log\big(p(I_{yr})\big),$$

where

$$p(I_{yr}) = \frac{|I_{yr}|}{\sum_{z=1}^{k^t} |I_{zr}|}.$$

$w(c_i, I_{ir}^n)$ represents the importance of node I_{ir} in cluster c_i with two factors, the probability of I_{ir} being in c_i and the weighting function $f(I_r)$. Based on the concepts of the node importance, the probability of I_{ir} being in c_i computes the frequency of I_{ir} in the cluster c_i, and the weighting function is designed to measure the distribution of the node between clusters based on the information theorem [28]. Entropy is the measurement of information and uncertainty on a random variable. Formally, if X is a discrete random variable with possible state $x_1, \ldots, x_n$ and $p(x_i) = \Pr(X = x_i)$ is the probability of the ith state of X, the entropy $E(X)$ is defined as shown in the following equation:

$$E(X) = -\sum_{i=1}^{n} p(x_i) \log(p(x_i)).$$

The weighting function $f(I_{ir}^n)$ measures the entropy of the node between clusters. The entropy $E(X)$ is maximal when the random variable X has a uniform distribution, which means that X possesses maximal uncertainty or minimum information when we obtain a value of X. The weighting function $f(I_{ir}^n)$ measures the entropy of the node between clusters. Suppose that there is a node that occurs in all clusters uniformly. The node that contains the maximum uncertainty provides less clustering characteristics. Therefore, this node should have a small weight. Moreover, the maximum entropy value of a node between clusters equals

Cluster c_1^1	
Node	Importance
A_1=A	1
A_2=M	0.029
A_3=C	0.67
A_3=D	0.33

Cluster c_2^1	
Node	Importance
A_1=X	0.5
A_1=Y	0.5
A_2=M	0.029
A_3=P	1

Fig. 3. The NIR of the clustering result C^1 in Fig. 2.

$\log k$. In order to normalize the weighting function from zero to one, the entropy value of a node between clusters is divided by $\log k$. After that, the normalized entropy is subtracted by one so that the node containing large entropy will obtain a small weight.

The importance of the node I_{ir} in cluster c_i is measured by multiplying the first concept, i.e., the probability of I_{ir} being in c_i, and the second concept, i.e., the weighting function $f(I_r)$. Note that the range of both the probability of I_{ir} being in c_i and the weighting function $f(I_{ir})$ is [0, 1], implying that the range of the important value $w(c_i, I_{ir})$ is also [0, 1].

NIR is related to the idea of conceptual clustering [11], which creates a conceptual structure to represent a concept (cluster) during clustering. However, NIR only analyzes the conceptual structure and does not perform clustering, i.e., there is no objective function such as category utility (CU) [15] in conceptual clustering to lead the clustering procedure. Furthermore, NIR *considers both the intracluster similarity and the intercluster similarity* in the representation by integrating the first and the second concepts.

Example 1. Consider the data set in Fig. 2. Cluster c_1^1 contains three data points. The node $\{[A_1 = A]\}$ occurs three times ($|I_{1,\{[A_1=A]\}}|=3$) in c_1^1 and does not occur in c_2^1. The weight of the node $f(I_{\{[A_1=A]\}}) = 1 - \frac{-1}{\log 2}(\frac{3}{3}\log\frac{3}{3} + \frac{0}{3}\log\frac{0}{3}) = 1$. Therefore, the importance of the node $\{[A_1 = A]\}$ in cluster c_1^1 is $w(c_1^1, \{[A_1 = A]\}) = 1 * \frac{3}{3} = 1$, and in cluster c_2^1, it is $w(c_2^1, \{[A_1 = A]\}) = 1 * 0 = 0$. In addition, the weight of the node $\{[A_2 = M]\}$, $f(I_{\{[A_2=M]\}}) = 1 - \frac{-1}{\log 2}(\frac{3}{5}\log\frac{3}{5} + \frac{2}{5}\log\frac{2}{5}) = 0.029$. Therefore, the importance of the node $\{[A_2 = M]\}$ in cluster c_1^1 is $w(c_1^1, \{[A_2 = M]\}) = 0.029 * \frac{3}{3} = 0.029$, and in cluster c_2^1, it is $w(c_2^1, \{[A_2 = M]\}) = 0.029 * \frac{2}{2} = 0.029$.

Finally, the *NIR* of cluster c_i can be represented as a table of the pairs $(I_{ir}, w(c_i, I_{ir}))$ for all the nodes in the cluster c_i. Fig. 3 shows the NIR of the clustering result C^1 in Fig. 2. For more details about NIR, please see our previous work in [7].

3 Drifting Concept Detection

In this section, we introduce our DCD algorithm. The objective of the DCD algorithm is to *detect the difference of cluster distributions between the current data subset S^t and the last clustering result $C^{[t_e,t-1]}$ and to decide whether the reclustering is required or not in S^t*. Therefore, the incoming categorical data points in S^t should be able to be allocated into the corresponding proper cluster at the last clustering result efficiently. We named the allocating process as "*data labeling*" [7]. In this paper, we modify our previous work on the labeling process in order to detect outliers in S^t. The data point that does not belong to any proper cluster is called an outlier. After labeling, the last clustering result $C^{[t_e,t-1]}$ and the current temporal clustering result C'^t obtained by data labeling are compared with each other. If the difference of cluster distributions is large enough, the sliding window t will be considered as a concept-drifting window, and S^t will perform reclustering. The flowchart of the DCD algorithm is shown in Fig. 4.

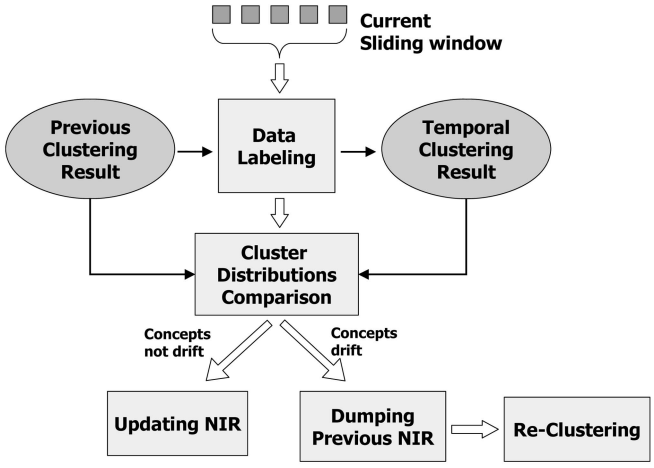


Fig. 4. The flowchart of the DCD algorithm.

In Section 3.1, we will introduce the data labeling process and the outlier detection. After that, the cluster distribution comparison method is presented in Section 3.2.

3.1 Data Labeling and Outlier Detection

The goal of data labeling is to *decide the most appropriate cluster label for each incoming data point*. Specifically, suppose that a data point p_j is given. The similarity $S(c_i, p_u)$ between p_j and cluster c_i, $1 \le i \le k$, is measured, and the cluster that obtains $\max(S(c_i, p_j))$ is considered as the most appropriate cluster. In this paper, the clusters are represented by an effective clustering representative, named NIR. Based on NIR, the similarity, referred to as *resemblance* in this paper, is defined below.

Definition 3 (resemblance and maximal resemblance). *Given a data point p_j and an NIR table of clusters c_i, the* resemblance *is defined by the following equation:*

$$R(p_j, c_i) = \sum_{r=1}^{q} w(c_i, I_{ir}),$$

where I_{ir} is one entry in the NIR table of clusters c_i.

The value of resemblance $R(p_j, c_i)$ can be directly obtained by summing up the nodes' importance in the NIR table of clusters c_i, where these nodes are decomposed from the data point p_j. This equation, which sums the nodes' importance, considers how much the data point is similar to the cluster based on the nodes in that data point. When a data point contains nodes that are more important in cluster c_x than in cluster c_y, $R(p_j, c_x)$ will be larger than $R(p_j, c_y)$.

Based on Definition 3, a data point p_j is labeled to the cluster that obtains the *maximal resemblance*. However, if a

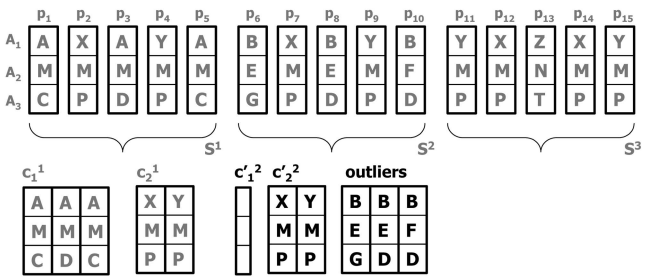


Fig. 5. The temporal clustering result C'^2 that is obtained by data labeling.

data point is not similar to any cluster in the last clustering result, all the resemblance values between each cluster will be small. In such a case, the data point will be treated as an outlier. Therefore, a threshold λ_i in each cluster is set to identify outliers. The decision function is defined as follows:

$$Label = \begin{cases} c_i^*, & \text{if } \max R(p_j, c_i) \geq \lambda_i, \text{ where } 1 \leq i \leq k, \\ outliers, & \text{otherwise.} \end{cases}$$

Since we measure the similarity between the data point p_j and the cluster c_i as $R(p_j, c_i)$, the cluster with the maximal resemblance is the most appropriate cluster for that data point. In addition, if the maximal resemblance (the most appropriate cluster) is smaller than the threshold λ_i in that cluster, the data point is seen as an outlier.

Example 2. Consider the data set in Fig. 2 and the NIR of C^1 in Fig. 3. The data points in the second sliding window are going to perform data labeling, and the thresholds $\lambda_1 = \lambda_2 = 0.5$. The first data point $p_6 = (B, E, G)$ in S^2 is decomposed into three nodes, i.e., $\{[A_1 = B]\}$, $\{[A_2 = E]\}$, and $\{[A_3 = G]\}$. The resemblance of p_6 in c_1^1 is zero, and in c_2^1, it is also zero. Since the maximal resemblance is not larger than the threshold, the data point p_6 is considered as an outlier. In addition, the resemblance of p_7 in c_1^1 is 0.029, and in c_2^1, it is 1.529 $(0.5 + 0.029 + 1)$. The maximal resemblance value is $R(p_7, c_2^1)$, and the resemblance value is larger than the threshold $\lambda_2 = 0.5$. Therefore, p_7 is labeled to cluster c_2^1.

The main difficulty on the decision function is to determine the values of thresholds λ_i. The simplest solution is to set a constant identical threshold for all clusters, i.e., $\lambda_1 = \lambda_2 = \cdots = \lambda_n = \lambda$. However, it is still hard to define a single value λ that is applied on all clusters to determine whether the data point is an outlier or not. Therefore, we use the data points in the last sliding window that construct the last clustering result to decide the thresholds λ_i. Since we know that each data point in the last sliding window belongs to whose cluster, the resemblance value between each data point and that cluster can be measured. Therefore, the smallest resemblance value in each cluster is set as λ_i. That is to say, the incoming data point is able to allocate to the cluster if the resemblance value is larger than the smallest resemblance value in that cluster measured by the data points in the last sliding window. For example, in the example in Figs. 2 and 3, $\lambda_1 = 1 + 0.029 + 0.33 = 1.359$, and $\lambda_2 = 0.5 + 0.029 + 1 = 1.529$. The temporal clustering result C'^2 is shown in Fig. 5.

After data labeling, the temporal clustering result that is based on the last clustering result on the current sliding window is obtained. Those two clustering results are compared with each other in the next step named the "*Cluster Distribution Comparison*" step. The drifting concept is detected when those two clustering results are quite different. In the next section, we will introduce how we compare two cluster distributions.

3.2 Cluster Distributions Comparison

In the Cluster Distribution Comparison step, the last clustering result and the current temporal clustering result obtained by data labeling are compared with each other to detect the drifting concept. The clustering results are said to be different according to the following two criteria:

1. *The clustering results are different if quite a large number of outliers are found by data labeling.*
2. *The clustering results are different if quite a large number of clusters are varied in the ratio of data points.*

Since the idea of data labeling is to present the original clustering characteristics to the incoming data points [7], the outliers that are not able to allocate to any cluster may be generated based on different concepts. Therefore, if too many outliers are detected by the data labeling step, the drifting concept may happen in the current sliding window. As a result, a threshold θ named *outlier threshold* is set in this step. If the ratio of outliers in the current sliding window is larger than the outlier threshold, the clustering results are said to be different, and the concept is said to drift.

Moreover, another type of drifting concept is also detected in this step. The ratio of data points in a cluster may be changed dramatically by a drifting concept, e.g., the cluster that contains half of the data points in the last clustering result has suddenly disappeared in the current clustering result. In order to detect the change, we adopt a double-threshold method. One threshold ϵ named *cluster variation threshold* is utilized to determine that the variation of the ratio of data points in a cluster is big enough. The cluster that exceeds the cluster variation threshold is seen as a different cluster. And then, the number of different clusters is counted, and the ratio of different clusters is compared with the other threshold η named *cluster difference threshold*. If the ratio of different clusters is larger than the cluster difference threshold, the concept is said to drift in the current sliding window. The entire Cluster Distribution Comparison equation is shown as follows:

$$\text{Concept drift} = \begin{cases} \text{Yes}, & \text{if } \frac{\#\text{ of outliers}}{N} > \theta \\ \text{Yes}, & \text{if } \frac{\sum_{i=1}^{k^{[t_e,t-1]}} d\left(c_i^{[t_e,t-1]}, c_i^{t}\right)}{k^{[t_e,t-1]}} > \eta, \\ & \text{where } d\left(c_i^{[t_e,t-1]}, c_i^{t}\right) \\ & = \begin{cases} 1, & \text{if } \left| \frac{m_i^{[t_e,t-1]}}{\sum_{x=1}^{k^{[t_e,t-1]}} m_x^{[t_e,t-1]}} - \frac{m_i^{t}}{\sum_{x=1}^{k^{[t_e,t-1]}} m_x^{t}} \right| > \epsilon \\ 0, & \text{otherwise} \end{cases} \\ \text{No}, & \text{otherwise} \end{cases}.$$

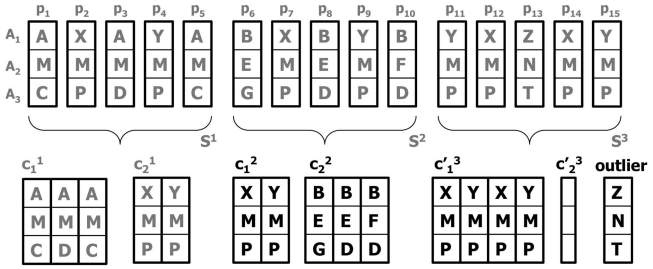


Fig. 6. The example of performing reclustering on S^2 and doing data labeling on S^3.

The ratio of outliers in the current sliding window t is first measured by this equation and compared with θ. After that, the variation of the ratio of data points in the cluster c_i between the last clustering result $C^{[t_e,t-1]}$ and the current temporal clustering result C'^t is calculated and compared by a zero-one function $d(c_i^{[t_e,t-1]}, c_i'^t)$, where the different cluster is represented by one. The number of different clusters is summed in this equation, and the ratio of different clusters between $C^{[t_e,t-1]}$ and C'^t is compared with η. If the current sliding window t considered that the drifting concept happens, the data points in the current sliding window t will perform reclustering. On the contrary, the current temporal clustering result is added into the last clustering result, and the clustering representative NIR is updated.

Example 3.1. Consider the example shown in Fig. 5. The last clustering result C^1 and current temporal clustering result C'^2 is compared with each other by the above equation. Suppose that the outlier threshold θ is set to 0.4, the cluster variation threshold ϵ is set to 0.3, and the cluster difference threshold η is set to 0.5. In Fig. 5, there are three outliers in C'^2, and the ratio of outliers in S^2 is $\frac{3}{5} = 0.6 > \theta$. Therefore, S^2 is considered as a concept-drifting window and is going to do reclustering.

Example 3.2. Moreover, the result of performing reclustering on S^2 and doing data labeling on S^3 is shown in Fig. 6. The above equation is also applied on the last clustering result C^2 and the current temporal clustering result C'^3. There is only one outlier in C'^3, and the ratio of outliers in S^3 is $\frac{1}{5} = 0.2 < \theta$. However, the variation of the ratio of data points between clusters c_1^2 and $c_1'^3$ is $|\frac{2}{5} - \frac{4}{5}| = 0.4 > \epsilon$, and also, the variation of the ratio of data points between clusters c_2^2 and $c_2'^3$ is $|\frac{3}{5} - \frac{0}{5}| = 0.6 > \epsilon$. Therefore, $d(c_1^2, c_1'^3) = d(c_2^2, c_2'^3) = 1$, and the number of different clusters is two. The ratio of different clusters between C^2 and C'^3 is $\frac{2}{2} = 1 > \eta$. Therefore, S^3 is also considered as a concept-drifting window and is going to perform reclustering. Finally, the reclustering result and the output NIR are shown in Fig. 7.

If the current sliding window t considered that the drifting concept happens, the reclustering process will be performed. The last clustering result $C^{[t_e,t-1]}$ represented in NIR is first dumped out with time stamp to show a steady clustering result that is generated by a stable concept from the last concept-drifting time stamp t_e to $t-1$. After that,

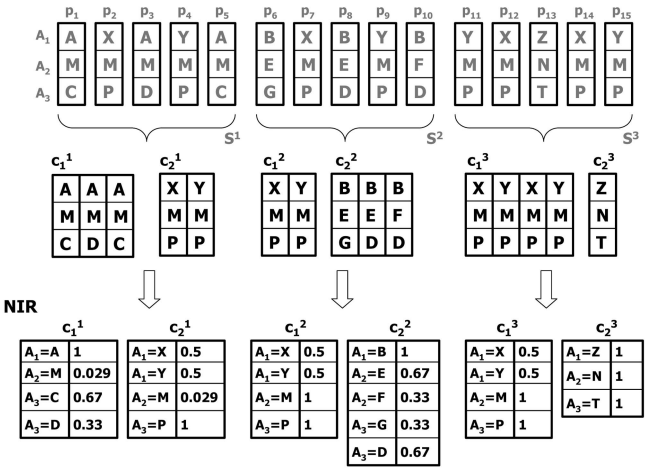


Fig. 7. The final clustering results on the example data set and the output NIR results.

the data points in the current sliding window t will perform reclustering, where the initial clustering algorithm is applied. The new clustering result C^t is also analyzed and represented by NIR. And finally, the data points in the next sliding window S^{t+1} and the clustering result C^t are input to do the DCD algorithm.

If the current sliding window t considered that the stable concept remained, the current temporal clustering result C'^t that is obtained from data labeling will be added into the last clustering result $C^{[t_e,t-1]}$ in order to fine-tune the current concept. In addition, the clustering representative NIR is also needed to be updated. For the reason of quickly updating the process, not only the importance but also the counts of each node in each cluster are recorded. Therefore, the count of the same node in $C^{[t_e,t-1]}$ and in C'^t is able to be summed directly, and the importance of each node in each of the merged clusters can be efficiently calculated by Definition 2.

3.3 Implementation of DCD

The DCD algorithm is shown in Algorithms 1 and 2. All the clustering results C are represented by NIR, which contains all the pairs of nodes and node importance. For better execution efficiency, the technique of *hashing* can be applied on the represented table, and the operation on querying the node importance have a time complexity of $O(1)$. Therefore, the resemblance value of the specific cluster is computed efficiently in Data Labeling shown in Algorithm 1 by the sum of each node importance through looking up the NIR hash table only q times, and the entire time complexity of data labeling is $O(q * k * N)$ [7]. In addition, in detecting drifting concepts, we compare the distribution between k clusters and check the ratio of outliers. All of the detecting criteria are effortless. The bottlenecks of the execution time in DCD may occur on the reclustering step when the concept drifts and on the updating NIR table step when the concept does not drift. When updating the NIR tables, we need to scan all of the nodes in the hash table and calculate their importance. However, when we do reclustering, the initial clustering algorithm is performed on S^t. The time complexity of most clustering algorithms is $O(N^2)$ and is extremely larger than the updating NIR table step.

Therefore, the real bottleneck of the execution time in DCD is to perform reclustering on the drifting-concept window.

Algorithm 1. DataLabeling($C^{[t_e,t-1]}, S^t$)
outliers $out = 0$
while there is next tuple in S^t **do**
read in data point p_j from S^t
divide p_j into nodes I_1 to I_q
for all clusters $c_i^{[t_e,t-1]}$ in $C^{[t_e,t-1]}$ **do**
calculate Resemblance $R(p_j, c_i^{[t_e,t-1]})$
end for
find Maximal Resemblance $c_m^{[t_e,t-1]}$
if $R(p_j, c_m^{[t_e,t-1]}) \geq \lambda_m$ **then**
p_j is assign to c_m^t
else
$out = out + 1$
end if
end while
return out

Algorithm 2 DriftingConceptDetecting($C^{[t_e,t-1]}, S^t$)
$outlier =$ **DataLabeling**($C^{[t_e,t-1]}, S^t$) {Do data labeling on current sliding window}
$numdiffclusters = 0$
for all clusters $c_i^{[t_e,t-1]}$ in $C^{[t_e,t-1]}$ **do**
if $\left| \frac{m_i^{[t_e,t-1]}}{\sum_{x=1}^{k^{[t_e,t-1]}} m_x^{[t_e,t-1]}} - \frac{m_i^t}{\sum_{x=1}^{k^{[t_e,t-1]}} m_x^t} \right| > \epsilon$ **then**
$numdiffclusters = numdiffclusters + 1$
end if
end for
if $\frac{outlier}{N} > \theta$ **or** $\frac{numdiffclusters}{k^{[t_e,t-1]}} > \eta$ **then**
{Concept Drifts}
dump out $C^{[t_e,t-1]}$
call initial clustering on S^t
else
{Concept not Drifts}
add C'^t into $C^{[t_e,t-1]}$
update NIR as $C^{[t_e,t]}$
end if

3.4 Setting System Parameters

In this framework, the system parameters, i.e., sliding window size and thresholds for detecting drifting concepts, may very for different application data. Several observations, which are presented in [12], provide the clues to set these parameters:

- Detecting small changes requires a low threshold value.
- Detecting dramatic changes requires a high threshold value.
- Detecting frequently occurring changes requires a short time frame.
- Detecting less frequent events requires long time frames.
- Detecting changes in unstable data sets requires a high threshold value.
- Detecting changes in stable data sets requires a low threshold value.

The similar results of the above observation are shown in our experimental results. Therefore, if we can obtain prior knowledge such as the frequency of the drifting concepts of the data from domain experts, the prior knowledge can help us to set proper parameter values.

Furthermore, if we have no idea about the application data, the following strategy is applied to help us to determine the parameter values. First, a fraction of data points in the front of the data set is selected and separated by a predefined window size. And then, the thresholds that are used to detect drifting concepts are calculated between adjacent windows. Finally, we can observe these values to find the proper threshold settings. In addition, the procedure can be applied to other window sizes to choose a proper window size.

4 Clustering Relationship Analysis

After we perform clustering on the entire data set D where the drifting concepts are considered, several clustering results with time stamps are obtained and represented by NIR. Each clustering result is generated from one concept that persists for a period of time. In addition to reporting each clustering result, we also provide the CRA algorithm, which tries to explain the drifting concepts based on the evolving clustering results. In [25], a similar strategy that analyzes the clustering results for mining evolving user profiles in the Web is proposed. This method defines the birth, persistence, atavism, and death of the profiles (clusters) to model the profiling Web usages. In order to deal with the Web data set, a particular hierarchical clustering algorithm is applied, and the algorithm for tracking evolving user profiles is based on the clustering results. In contrast, in this paper, we propose a generalized framework that detects the drifting concept and try to show the evolving clustering results in the categorical domain.

CRA measures the similarity of clusters between the clustering results at different time stamps and links the similar clusters. The linked clusters show the relationship from the last clustering result to the current result. Explicitly, the linking situation presents that the clusters in the current clustering result are split, merged, or changed in size from the last clustering result. Based on the relationship analysis, the *evolving clusters* will provide clues for us to catch the time-evolving trends in the data set. In the following, we will introduce the main problem in CRA—*how to link similar clusters at different time stamps*, i.e., how to calculate the similarity between each pair of clusters at different time stamps.

4.1 Node Importance Vector and Cluster Distance

In this framework, the clustering results are reported by the clustering representative NIR. Another view of this representative is to see the domain of nodes as a *vector space*. The space consists of all nodes in the entire data set, i.e., the bases of this space are all nodes that occur in the entire data set. A cluster in this space is a *vector*, and each entry in this vector is the value of importance in that node domain. Based on the node vector space, the node importance vector of a cluster c_i is defined as follows:

$\overline{c_1^1}$ =(1, 0, 0, 0, 0, 0, 0, 0.029, 0, 0.67, 0.33, 0, 0, 0)
$\overline{c_2^1}$ =(0, 0, 0.5, 0.5, 0, 0, 0, 0.029, 0, 0, 0, 0, 1, 0)
$\overline{c_1^2}$ =(0, 0, 0.5, 0.5, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0)
$\overline{c_2^2}$ =(0, 1, 0, 0, 0, 0.67, 0.33, 0, 0, 0, 0.67, 0.33, 0, 0)

Fig. 8. The node importance vectors $\overline{c_1^1}$, $\overline{c_2^1}$, $\overline{c_1^2}$, and $\overline{c_2^2}$ of the clustering results C^1 and C^2 in Fig. 7.

Definition 4 (node importance vector). *Suppose that there are totally z distinct nodes in the entire data set D. The node importance vector $\overline{c_i}$ of a cluster c_i is defined as the following equation:*

$$\overline{c_i} = (W_i(I_1), W_i(I_2), \ldots, W_i(I_r), \ldots, W_i(I_z)),$$

where

$$\begin{cases} W_i(I_r) = 0, & \text{if } I_r \text{ does not occur in } c_i, \\ W_i(I_r) = w(c_i, I_{ir}), & \text{if } I_r \text{ occurs in } c_i. \end{cases}$$

The value in the vector $\overline{c_i}$ on each node domain is the importance value of this node in cluster c_i, i.e., $w(c_i, I_{ir})$. If the node does not occur in cluster c_i, the value in the vector $\overline{c_i}$ on this node domain is zero. Note that the domain of attribute values contains all distinct nodes that occur in the entire data set, not just in cluster c_i. Therefore, *the dimensions of all the vectors $\overline{c_i}$ are the same.*

Example 4. In the example data set shown in Fig. 2, there are totally 14 distinct nodes in the entire data set, and the NIR results of C^1 and C^2 are shown in Fig. 7. Suppose that the vector space in this example is defined by the following 14 nodes: $([A_1=A], [A_1=B], [A_1=X], [A_1=Y], [A_1=Z], [A_2=E], [A_2=F], [A_2=M], [A_2=N], [A_3=C], [A_3=D], [A_3=G], [A_3=P], [A_3=T])$. The vector of cluster c_1^1 is $\overline{c_1^1}=(1,0,0,0,0,0,0,0.029,0,0.67,0.33,0,0,0)$. Fig. 8 shows four node importance vectors in the clustering results C^1 and C^2.

Based on the foregoing, clusters c_i and c_j are represented by the node importance vectors $\overline{c_i}$ and $\overline{c_j}$. We define the similarity between clusters c_i and c_j by the *cosine measure* [26] between vectors $\overline{c_i}$ and $\overline{c_j}$. Cosine measure, which calculates the cosine of the angle between two vectors, is a popular measure of similarity in the vector space model [27]. The cosine measure between vectors $\overline{c_i}$ and $\overline{c_j}$ is calculated as the following equation:

$$CM(\overline{c_i}, \overline{c_j}) = \frac{\sum_{r=1}^{z} W_i(I_r) * W_j(I_r)}{\sqrt{\sum_{r=1}^{z} W_i(I_r)^2}\sqrt{\sum_{r=1}^{z} W_j(I_r)^2}}.$$

Example 5. Consider the clustering results C^1 and C^2 in Fig. 7. The node importance vectors of the clustering results C^1 and C^2 are shown in Fig. 9. The similarity between vectors $\overline{c_1^1}$ and $\overline{c_1^2}$ is

$$CM\left(\overline{c_1^1}, \overline{c_1^2}\right) = \frac{(1*0)+(0*0.5)+(0*0.5)+(0.029*1)+(0.67*0)+(0.33*0)+(0*1)}{\sqrt{1^2+0.029^2+0.67^2+0.33^2}\sqrt{0.5^2+0.5^2+1^2+1^2}}$$
$$= \frac{0.029}{1.248*1.581} = 0.015.$$

	Cluster c_1^2	Cluster c_2^2
Cluster c_1^1	0.015	0.122
Cluster c_2^1	0.789	0
	Cluster c_1^3	Cluster c_2^3
Cluster c_1^2	1	0
Cluster c_2^2	0	0

Fig. 9. The similarity table between the clustering results C^1 and C^2 and between the clustering results C^2 and C^3 in Fig. 7.

In addition, $CM(\overline{c_2^1}, \overline{c_1^2}) = \frac{1.529}{1.225*1.581} = 0.789$, which is larger than $CM(\overline{c_1^1}, \overline{c_1^2})$. Therefore, cluster c_2^1 is said to be more similar to cluster c_1^2 than to cluster c_1^1.

In CRA, the similarity of each pair of adjacent clustering results, i.e., $C^{[t_a,t_b-1]}$ and $C^{[t_b,t_c]}$, where t_b is the time stamp that drifting concept happens, is measured by the cosine measure. All the similarities between clusters $c_i^{[t_a,t_b-1]}$ and $c_j^{[t_b,t_c]}$, where $c_i^{[t_a,t_b-1]}$ is in $C^{[t_a,t_b-1]}$ and $c_j^{[t_b,t_c]}$ is in $C^{[t_b,t_c]}$, are computed, and the similarities form a similarity table. Fig. 9 shows the similarity tables between clustering results C^1 and C^2, and between clustering results C^2 and C^3. We may set a similarity threshold to link similar clusters such that the similarity between these two clusters is higher than the threshold. In order to show the relationship between clustering results, a *visualizing method* is devised in this paper to facilitate the observation of the evolving clusters.

4.2 Visualizing the Evolving Clusters

Fig. 10 shows our visualizing method. In the upper area for *showing evolving clusters*, several circles in a column indicate a clustering result, and the horizontal direction is the time axis. Note that the size of each circle is different. It is because *the ratio of data points in the clustering result* is represented by the size of circle. Moreover, if there are too many clusters in the clustering results, users can set up a proportion threshold such that only clusters where the ratio of data points is larger than the user threshold appear in the area showing evolving clusters.

Additionally, in order to display the relationship between clusters at different time stamps, we use *lines with different sizes* to link the similar clusters, and the size of a

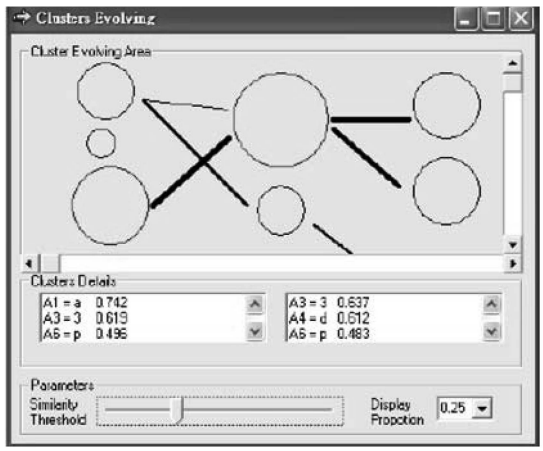


Fig. 10. The visualization of the evolving clusters.

line indicates the intensity of the similarity between clusters, which is computed by the cosine measure.

However, the similarities between each pair of clusters are recorded in the table that is generated by CRA, meaning that each pair of clusters is linked by a line. If we mark all cluster relations in the visualization, *the lines are too many to show the clusters evolving*. Therefore, a user-specified threshold is used to prune the unimportant relationship when the similarity between clusters is too small. In this visualization, a slider is provided for the user to select the user-specified threshold, and the pruning actions will display immediately on the area showing evolving clusters.

Moreover, in the middle of the visualization, the *details of a cluster* are shown in this area. If a cluster is selected, the nodes and the importance of nodes are reported in this area. There are two areas to show the details of clusters, and users can hence compare two clusters by the distribution of nodes. In addition, users can select several clusters to be the observing targets. The visualization will show the relationship only from those observing targets. Therefore, users can easily track some evolving trends from the clusters that are interesting to users.

5 Experimental Results

In this section, we demonstrate the scalability and accuracy of the framework on clustering the evolving categorical data. In Section 5.1, the test environment and the data sets used are described. Section 5.2 presents the efficiency and the scalability of DCD, and Section 5.3 presents the accuracy on the evaluation result of DCD.

5.1 Test Environment

The experiments are conducted on a PC with an Intel Pentium 4 3.2-GHz processor and 1-Gbyte memory running the Windows XP SP2 operating system. In all experiments, the clustering algorithm, *EM* [10], is chosen to do the initial clustering and reclustering on the data sets. We compare our framework on both scalability and accuracy evaluations with performing clustering algorithm EM once on the entire data set. In the following, the synthetic and real data sets that are utilized in the experiments are described.

Synthetic data sets. In this paper, we utilize the clustering data generator that is discussed in [34], and the generator is extended to synthesize a clustering data set with multidimensions. However, the generator only produces numerical data. In order to obtain the clustering data in the categorical domain, *uniform quantification* is performed on each dimension after the numerical data set is generated. Therefore, we will obtain the categorical data points and the clustering label that each data point belongs to by this data generator. In addition, in order to simulate data sets with drifting concepts, we create several different clustering results that are different in the number of clusters and the centroid location of the clusters but the same in the dimensionality, by the above data generator. A drifting concept is generated by combining two different clustering results. We randomly select and combine the clustering results several times to obtain a data set with drifting concepts. The detail setting of synthetic data sets utilized in the experiments are shown in Table 2.

TABLE 2
The Parameter Setting of the Synthetic Data Sets Utilized in the Experiments

Settings	# of C	k in each C	q	# of data points in each C	# of values in A_a
D_1	5	5 ~10	3	10000 $\pm$1%	10
D_2	5	3 ~10	3	3000 ~10000 $\pm$1%	10
D_3	10	5 ~20	10	10000 $\pm$1%	20
D_4	10	5 ~40	20	3000 ~10000 $\pm$1%	20

In setting D_1, we focus on varying the number of clusters in different clustering results where the difference is caused by drifting concepts. The drifting concepts occur once per around 10,000 data points stably. In setting D_2, the number of data points in each clustering result is also varied. Therefore, the drifting concepts do not occur regularly. In settings D_3 and D_4, we try to establish large-scale testing data sets where the range of cluster numbers is enlarged and the data dimensionality is increased.

Real data set. We employed the *KDD-CUP'99 Network Intrusion Detection data set* for our experiments. The Network Intrusion Detection data set consists of a series of TCP connection records from two weeks of LAN traffic managed by MIT Lincoln Labs. Each record can correspond to either a normal connection or an intrusion or attack. The attack types are classified into one of 24 types such as buffer-overflow, guess-passwd, neptune, smurf, warezclient, and so on. Most of the connections in this data set are normal, but occasionally, there could be a burst of attacks at certain times. One of the objectives in the intrusion detection system is to detect the changes of connections from normal to a burst of attacks or from the attacks back to normal, and those changes naturally correspond to a drifting concept. Therefore, this data set is a time-evolving data and is suitable for evaluating our algorithm.

In this experiment, we utilize the 10 percent subset version that is provided from the KDD-CUP '99 Website. In this data set, there are 493,857 records, and each record contains 42 attributes (class label is not included) such as the duration of the connection, the number of data bytes transmitted from the source to the destination (and vice versa), the number of "root" accesses, etc. And also, 34 attributes are continuous. We adopt uniform quantization on those numerical attributes where each attribute is quantized into five categorical values.

This real data set is applied on the accuracy evaluation. We utilize the class label that indicates that the record is a normal connection or an attack to identify the drifting concept. All 24 different attack types are seen as "attack." A drifting concept is recognized if the change is continued for at least 300 connections. For example, the first 7,793 records in this data set are normal. And then, 3,695 smurf attacks come after the normal connections. Therefore, this change is marked as a drifting concept. Based on the above criterion, 33 drifting concepts are identified in the intrusion detection data set.

5.2 Evaluation on Efficiency and Scalability

Fig. 11 shows the scalability with the data size of DCD. This study fixes the dimensionality to 20 and the number of

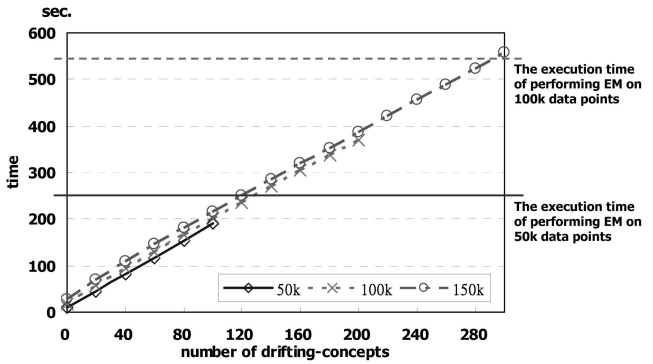


Fig. 11. Execution time comparison: scalability with data size and the number of drifting concepts.

clusters to 20 and also tests DCD in different numbers of data points, i.e., 50,000, 100,000, and 150,000. The sliding window size is set to 500. For the reason that the bottleneck of the execution time in DCD is to perform reclustering on the concept-drifting window, the number of drifting concepts directly impacts the execution time of DCD. The result is, shown in Fig. 11, that as the number of drifting concepts increases, the execution time of DCD also increases. Moreover, the increase of the data size only has little influence on the execution time when the number of drifting concepts is the same. In addition, even if all the windows are detected as the concept-drifting windows, the execution time of DCD is still faster than that of EM. Therefore, algorithm DCD can ensure efficient execution on clustering time-evolving data when the data size is large.

The scalability of DCD with data dimensionality and the number of clusters are shown in Fig. 12. In Fig. 12a, the study fixes the data size to 50,000 and the number of cluster to 20 and also varies the data dimensionality q as 20, 30, and 50. In Fig. 12b, the study fixes the data size to 50,000 and the data dimensionality to 20 and also varies the number of clusters k as 20, 30, and 50. Both studies set the sliding window size to 500. The experiments also show that the number of drifting concepts that require doing reclustering is the bottleneck of the execution time in DCD when the data dimensionality and the number of clusters are varied. However, the execution time of DCD is still faster than that of EM regardless of how many drifting concepts happen, showing the robustness of the DCD algorithm.

5.3 Evaluation on Accuracy

In this experiment, we test the accuracy of DCD on both synthetic and real data sets. First, we will test the accuracy of drifting concepts that are detected by DCD. And then, in order to evaluate the results of clustering algorithms, we adopt the following two widely used methods.

The CU function. The CU function [15] attempts to maximize both the probability that two data points in the same cluster obtain the same attribute values and the probability that data points from different clusters have different attributes. The expression to calculate the expected value of the CU function is shown in the following equation:

$$CU = \sum_{i=1}^{k} \frac{m_i}{N} \sum_{r=1}^{z} \Big[P(I_r|c_i)^2 - P(I_r)^2 \Big],$$

where the number of data points in cluster c_i is m_i, and there are totally z distinct nodes in the clustering results.

Confusion matrix accuracy (CMA). Since the synthetic data sets shown in Table 2 contain the clustering label on each data point, we can evaluate the clustering results by comparing with the original clustering labels. In the confusion matrix [2], the entry $(i; j)$ is equal to the number of data points assigned to output cluster c_i and that contain the original clustering label j. We measure the accuracy in this matrix (CMA) by maximizing the count of the one-to-one mapping in which one output cluster c_i is mapped to one original clustering label j.

5.3.1 Accuracy Evaluation on Synthetic Data Set

Based on the settings of synthetic data sets shown in Table 2, we can randomly select and combine the clustering results several times from one setting to obtain the data sets with drifting concepts and then test the detecting accuracy of algorithm DCD by those data sets. Table 3 shows the precision and recall of DCD on detecting drifting concepts. The outlier threshold θ is set to 0.1, and the cluster variation threshold ϵ is set to 0.1, and also, the cluster difference threshold η is set to 0.5. The number of clusters k, which is

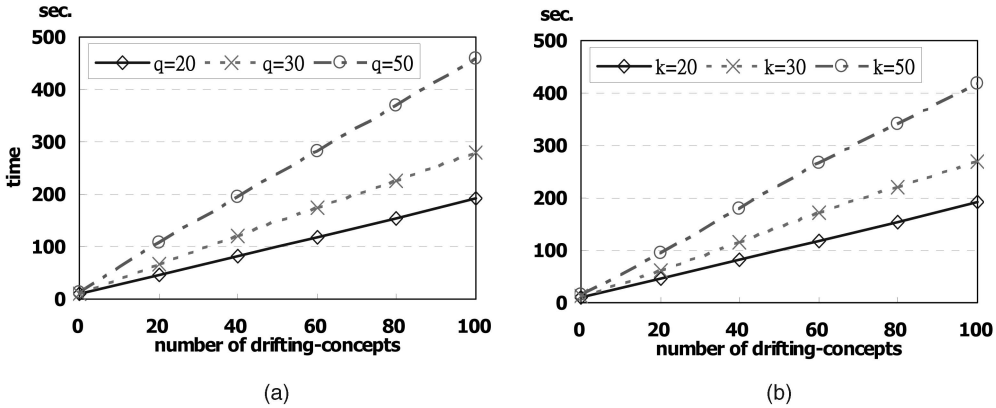


Fig. 12. Execution time comparison: scalability with data dimensionality and the number of clusters. (a) Scalability with data dimensionality. (b) Scalability with number of clusters.

TABLE 3
The Precision and Recall of DCD on Detecting Drifting Concepts on Synthetic Data Sets: Varying the Sliding Window Size N

Settings	# of drifting-concepts	$N = 1000$		$N = 2000$		$N = 3000$	
		Precision	Recall	Precision	Recall	Precision	Recall
D_1	37.8	0.667	0.973	0.814	0.946	0.857	0.919
D_2	38.2	0.755	0.974	0.878	0.947	0.895	0.868
D_3	48.85	0.716	0.98	0.839	0.959	0.87	0.918
D_4	47.6	0.643	0.968	0.824	0.933	0.884	0.902

the required parameter on the initial clustering step and reclustering step, is set to the maximum number of clusters in each setting, e.g., $k = 10$ in D_1 and $k = 20$ in D_3. In addition, each synthetic data set is generated by randomly combining 50 clustering results on that data set setting, and the precision and recall shown in Table 3 are the averages of 20 experiments.

The experimental results shown in Table 3 demonstrate that DCD is effective for detecting drifting concepts. The precision and recall are more than 80 percent when the size of the sliding window is larger than 2,000. The detecting precision is a little low when the size of the sliding window is set to 1,000 because the drifting concepts often cross two windows. In such a case, two windows are detected as drifting concepts by DCD, but we only count one as a correct hit, and the other window is considered as a miss. However, the detecting recall is the highest one when the size of sliding window is set to 1,000. The result can be explained by the reason that the drifting concepts will probably not be omitted in the sliding window when the data set is separated in detail. Based on this experiment, the size of the sliding window actually influences the effectiveness of DCD. If the data set varies dramatically, we can set a smaller sliding window size to capture the frequent drifting concepts. On the contrary, if the data set is stable, the size of the sliding window can be set larger in order to save the execution time.

In addition, we also evaluate the clustering results that are generated by DCD and compare with performing EM clustering once on the entire data set. In order to demonstrate the ability of DCD to cluster time-evolving categorical data, we pick two example data sets that are synthesized by settings D_1 and D_2 and evaluate the clustering result on each sliding window by CU and CMA. Fig. 13 shows the variations of CU and CMA between DCD and EM from the first sliding window to the last one on the example data set synthesized by D_1. The thresholds are set the same as the above experiment, and the sliding window size is set to 2,000. In this data set, the drifting concepts occur once per five sliding windows. In Fig. 13, the algorithm DCD outperforms EM on both CU and CMA, and the variation of CU and CMA on doing EM once is quite larger than DCD. The result can be explained by the reason that doing EM once on the entire data set cannot fit all the concepts. On the contrary, algorithm DCD, which generates a new clustering result when the drifting concept is detected, quickly responds to the trend of the evolving data set.

Fig. 14 shows the variations of CU and CMA between DCD and EM on the example data set synthesized by D_2. In this data set, the drifting concepts occur irregularly. In Fig. 14, the algorithm DCD also outperforms EM on both CU and CMA. Consequently, according to the observations on these two examples, the DCD algorithm generates better clustering results than performing EM once on the entire data set when we do clustering on the categorical time-evolving data.

5.3.2 Accuracy Evaluation on Real Data Set

We evaluate the DCD algorithm by the KDD-CUP '99 Network Intrusion Detection data set. The result is shown in Table 4. We recognize 33 drifting concepts in which the connection logs are changed either from normal to a burst of attacks or from attacks to normal. In this experiment, the

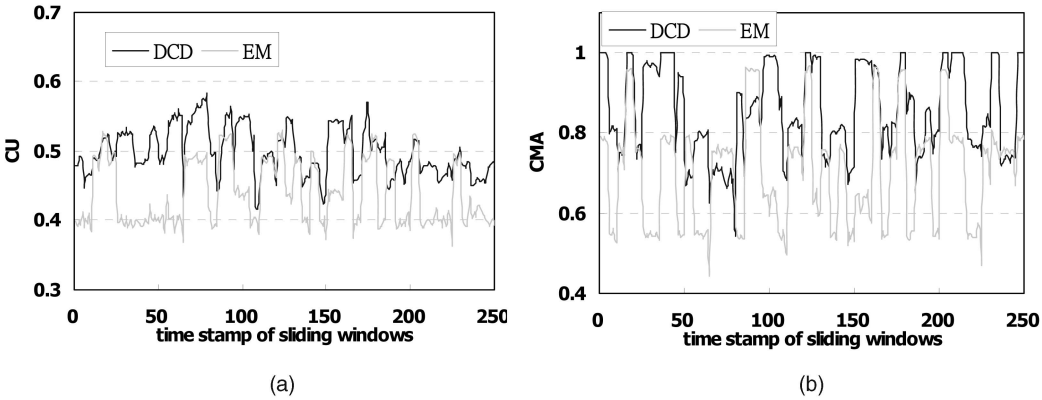
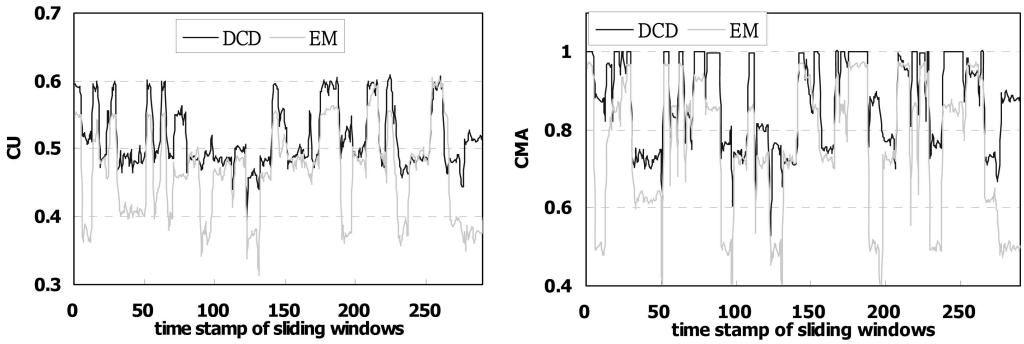


Fig. 13. The variations of CU and CMA between DCD and EM on the example data set synthesized by D_1. (a) Comparison of CU. (b) Comparison of CMA.



(a) (b)

Fig. 14. The variations of CU and CMA between DCD and EM on the example data set synthesized by D_2. (a) Comparison of CU. (b) Comparison of CMA.

TABLE 4
The Precisions and Recalls of DCD on Detecting Drifting Concepts on a Real Data Set: Varying the Sliding Window Size N

Data Set	number of drifting-concepts	$N = 1000$		$N = 2000$		$N = 3000$	
		Precision	Recall	Precision	Recall	Precision	Recall
KDD CUP99	33	0.604 (32/53)	0.97 (32/33)	0.769 (30/39)	0.91 (30/33)	0.903 (28/31)	0.838 (28/33)

outlier threshold θ is set to 0.1, and the cluster variation threshold ϵ is set to 0.1, and also, the cluster difference threshold η is set to 0.5. The number of clusters k, which is utilized in the initial clustering step and reclustering step, is set to 10. In Table 4, the result is similar to the synthetic data sets where the small sliding window size is induced to a high recall but a little low precision. In this case, the data set does not evolve frequently, i.e., most of the connections are normal but occasionally with a burst of attacks. Therefore, a larger sliding window size may be better suited than a smaller sliding window size.

In addition, we also evaluate the clustering results generated by DCD and compare with performing EM once on the entire data set. Fig. 15 shows the comparison of CU between DCD and EM on each sliding window. The sliding window size is set to 3,000. In Fig. 15, the values of CU drop to zero in the range of 51-114, 134-149, and 155-160 because the records are the same in those sliding windows, and the clustering results on these time stamps only contain an individual cluster. The peak value of CU in DCD is the time stamp that a drifting concept occurs. It is clear that the clustering results generated by DCD in each sliding window are better than doing EM once on the entire data set, especially when a drifting concept happened. Therefore, DCD is able to quickly reflect the drifting concept and generate better clustering results.

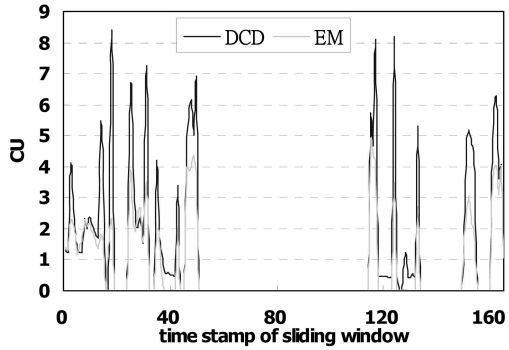


Fig. 15. The variations of CU between DCD and EM on the Network Intrusion Detection data set.

6 Related Work

In this section, we briefly review several numerical clustering algorithms that consider the time-evolving data and traditional categorical clustering algorithms.

The problem of clustering time-evolving data in the numerical domain has been explored in previous works [1], [5], [6], [8], [9], [12], [24], [32]. The idea is first introduced in [1], which views the data set to continuously evolve with time. The snapshots of microclustering results over the landmark window are first stored, and the clustering results over user-defined time periods are generated with flexibility. In [5], a density-based method for discovering arbitrary shape clusters in an evolving data set is proposed. Based on the concept of density, a core-micro-cluster is used to summarize the clusters in each sliding window instead of the original snapshots of microclustering results. In [24], a density-based method that takes the emergence of new patterns, the forgetting of old patterns, and the presence of the outliers into consideration is presented. This approach is focused on performing robust clustering with high quality on noisy and evolving data streams.

In [6] and [8], another view of clustering time-evolving data in the numerical domain, namely, "evolutionary clustering," is proposed. Evolutionary clustering performs data clustering over time and tries to optimize two potentially conflicting criteria: first, a cluster should be similar to the cluster at the previous time step (without drifting concept), and second, clustering should reflect the

data arriving during that time step (with drifting concept). In [6], the problem of evolutionary clustering is formulated, and two widely used clustering algorithms, K-Means and agglomerative hierarchical clustering, are extended to the evolutionary version. In [8], the idea of "temporal smoothness" is proposed based on the assumption that current clusters do not deviate dramatically from the most recent history clustering. Based on the idea, the framework for evolutionary spectral clustering in which the temporal smoothness is incorporated into the overall clustering quality is proposed.

In [12], a similar strategy with our approach that utilizes clustering results to analyze the drifting concepts is proposed. Several numerical characteristics such as the mean and standard deviation of cluster centers are used to represent clustering results and detect the changes in the data stream. After the change is detected, an offline voting-based classification algorithm is performed to associate each change with its correspondent event.

In [25], a similar strategy that analyzes the clustering results for mining evolving user profiles in the Web is proposed. This method defines the birth, persistence, atavism, and death of the profiles (clusters) to model the profiling Web usages. In order to deal with the Web data set, a particular hierarchical clustering algorithm is applied, and the algorithm for tracking evolving user profiles is based on the clustering results. In contrast, in this paper, we propose a generalized framework that detects the drifting concept and try to show the evolving clustering results in the categorical domain.

Based on the foregoing, the problem of clustering time-evolving data in the numerical domain has been explored and discussed in several previous works. However, in the categorical domain, this problem has not received explicit consideration. Therefore, in this work, we focus on the problem of clustering time-evolving categorical data and show the drifting concepts by the evolving clustering results.

The problem of clustering categorical data is first addressed in [17], which performs clustering on customer transaction data in a market database. In STIRR [14], the categorical clustering problem was mapped to a type of nonlinear dynamic systems. If the dynamic system converges, the categorical data set can be clustered. K-modes [19] extends the famous clustering algorithm K-Means to the categorical domain. A cluster is represented by "mode," which is composed by the most frequent attribute value in each attribute domain in this cluster. Several extension works based on k-modes are presented for different objectives, e.g., fuzzy k-modes [20], initial points refinement [29], etc. ROCK [16], an agglomerative hierarchical clustering algorithm, merges data points and clusters according to the sum of neighbors between clusters. In algorithms CACTUS [13] and CLICK [33], the basic idea behind these algorithms is to calculate the co-occurrence for attribute-value pairs. And then, the cluster is composed of the attribute values with high co-occurrence. In statistical categorical clustering algorithms such as COOLCAT [4] and LIMBO [3], data points are grouped based on the statistics. In algorithm COOLCAT [4], data points are separated in such a way that the expected entropy of the whole arrangement is minimized. In algorithm LIMBO [3], the Information Bottleneck method is applied to minimize the information lost, which resulted from summarizing data points into clusters.

However, all of the above categorical clustering algorithms focus on performing clustering on the entire data set and do not consider the time-evolving trends. In recent years, due to the increasing use of the record-based databases where data are being continuously added, enormous amounts of data are accumulated at a rapid pace. The problem of considering the current trend presented by newly incoming data is an important issue and has attracted much attention in recent years. In this paper, we address the problem of clustering time-evolving categorical data and devise an effective framework for detecting and analyzing the drifting concepts.

7 Conclusions

In this paper, we proposed a framework to *perform clustering on categorical time-evolving data.* The framework detects the drifting concepts at different sliding windows, generates the clustering results based on the current concept, and also shows the relationship between clustering results by visualization. In order to detect the drifting concepts at different sliding windows, we proposed the algorithm DCD to compare the cluster distributions between the last clustering result and the temporal current clustering result. If the results are quite different, the last clustering result will be dumped out, and the current data in this sliding window will perform reclustering. In addition, in order to observe the relationship between different clustering results, we proposed the algorithm CRA to analyze and show the changes between different clustering results. The experimental evaluation shows that performing DCD is faster than doing clustering once on the entire data set, and DCD can provide high-quality clustering results with correctly detected drifting concepts in both synthetic and real data cases. Therefore, the result demonstrates that our framework is practical for detecting drifting concepts in time-evolving categorical data.

Acknowledgments

The work was supported in part by the National Science Council of Taiwan, ROC., under Contract NSC95-2752-E-002-006-PAE.

References

[1] C. Aggarwal, J. Han, J. Wang, and P. Yu, "A Framework for Clustering Evolving Data Streams," *Proc. 29th Int'l Conf. Very Large Data Bases (VLDB),* 2003.

[2] C.C. Aggarwal, J.L. Wolf, P.S. Yu, C. Procopiuc, and J.S. Park, "Fast Algorithms for Projected Clustering," *Proc. ACM SIGMOD '99,* pp. 61-72, 1999.

[3] P. Andritsos, P. Tsaparas, R.J. Miller, and K.C. Sevcik, "Limbo: Scalable Clustering of Categorical Data," *Proc. Ninth Int'l Conf. Extending Database Technology (EDBT),* 2004.

[4] D. Barbará, Y. Li, and J. Couto, "Coolcat: An Entropy-Based Algorithm for Categorical Clustering," *Proc. ACM Int'l Conf. Information and Knowledge Management (CIKM),* 2002.

[5] F. Cao, M. Ester, W. Qian, and A. Zhou, "Density-Based Clustering over an Evolving Data Stream with Noise," *Proc. Sixth SIAM Int'l Conf. Data Mining (SDM),* 2006.
[6] D. Chakrabarti, R. Kumar, and A. Tomkins, "Evolutionary Clustering," *Proc. ACM SIGKDD '06,* pp. 554-560, 2006.
[7] H.-L. Chen, K.-T. Chuang, and M.-S. Chen, "Labeling Unclustered Categorical Data into Clusters Based on the Important Attribute Values," *Proc. Fifth IEEE Int'l Conf. Data Mining (ICDM),* 2005.
[8] Y. Chi, X.-D. Song, D.-Y. Zhou, K. Hino, and B.L. Tseng, "Evolutionary Spectral Clustering by Incorporating Temporal Smoothness," *Proc. ACM SIGKDD '07,* pp. 153-162, 2007.
[9] B.-R. Dai, J.-W. Huang, M.-Y. Yeh, and M.-S. Chen, "Adaptive Clustering for Multiple Evolving Streams," *IEEE Trans. Knowledge and Data Eng.,* vol. 18, no. 9, pp. 1166-1180, Sept. 2006.
[10] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *J. Royal Statistical Soc.,* 1977.
[11] D.H. Fisher, "Knowledge Acquisition via Incremental Conceptual Clustering," *Machine Learning,* 1987.
[12] M.M. Gaber and P.S. Yu, "Detection and Classification of Changes in Evolving Data Streams," *Int'l J. Information Technology and Decision Making,* vol. 5, no. 4, pp. 659-670, 2006.
[13] V. Ganti, J. Gehrke, and R. Ramakrishnan, "CACTUS—Clustering Categorical Data Using Summaries," *Proc. ACM SIGKDD,* 1999.
[14] D. Gibson, J.M. Kleinberg, and P. Raghavan, "Clustering Categorical Data: An Approach Based on Dynamical Systems," *VLDB J.,* vol. 8, nos. 3-4, pp. 222-236, 2000.
[15] M.A. Gluck and J.E. Corter, "Information Uncertainty and the Utility of Categories," *Proc. Seventh Ann. Conf. Cognitive Science Soc.,* pp. 283-287, 1985.
[16] S. Guha, R. Rastogi, and K. Shim, "ROCK: A Robust Clustering Algorithm for Categorical Attributes," *Proc. 15th Int'l Conf. Data Eng. (ICDE),* 1999.
[17] E.-H. Han, G. Karypis, V. Kumar, and B. Mobasher, "Clustering Based on Association Rule Hypergraphs," *Proc. ACM SIGMOD Workshop Research Issues in Data Mining and Knowledge Discovery (DMKD),* 1997.
[18] J. Han and M. Kamber, *Data Mining: Concepts and Techniques.* Morgan Kaufmann, 2001.
[19] Z. Huang, "Extensions to the *k*-Means Algorithm for Clustering Large Data Sets with Categorical Values," *Data Mining and Knowledge Discovery,* 1998.
[20] Z. Huang and M.K. Ng, "A Fuzzy *k*-Modes Algorithm for Clustering Categorical Data," *IEEE Trans. Fuzzy Systems,* 1999.
[21] G. Hulten, L. Spencer, and P. Domingos, "Mining Time-Changing Data Streams," *Proc. ACM SIGKDD,* 2001.
[22] A. Jain and R. Dubes, *Algorithms for Clustering Data.* Prentice Hall, 1988.
[23] A.K. Jain, M.N. Murty, and P.J. Flynn, "Data Clustering: A Review," *ACM Computing Surveys,* 1999.
[24] O. Nasraoui and C. Rojas, "Robust Clustering for Tracking Noisy Evolving Data Streams," *Proc. Sixth SIAM Int'l Conf. Data Mining (SDM),* 2006.
[25] O. Nasraoui, M. Soliman, E. Saka, A. Badia, and R. Germain, "A Web Usage Mining Framework for Mining Evolving User Profiles in Dynamic Web Sites," *IEEE Trans. Knowledge and Data Eng.,* vol. 20, no. 2, pp. 202-215, Feb. 2008.
[26] G. Salton and M.J. McGill, *Introduction to Modern Information Retrieval.* McGraw-Hill, 1986.
[27] G. Salton, A. Wong, and C.S. Yang, "A Vector Space Model for Automatic Indexing," *Comm. ACM,* vol. 18, no. 11, pp. 613-620, 1975.
[28] C.E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical J.,* 1948.
[29] Y. Sun, Q. Zhu, and Z. Chen, "An Iterative Initial-Points Refinement Algorithm for Categorical Data Clustering," *Pattern Recognition Letters,* vol. 23, no. 7, 2002.
[30] H. Wang, W. Fan, P. Yun, and J. Han, "Mining Concept-Drifting Data Streams Using Ensemble Classifiers," *Proc. ACM SIGKDD,* 2003.
[31] G. Widmer and M. Kubat, "Learning in the Presence of Concept Drift and Hidden Contexts," *Machine Learning,* 1996.
[32] M.-Y. Yeh, B.-R. Dai, and M.-S. Chen, "Clustering over Multiple Evolving Streams by Events and Correlations," *IEEE Trans. Knowledge and Data Eng.,* vol. 19, no. 10, pp. 1349-1362, Oct. 2007.
[33] M.J. Zaki and M. Peters, "Clicks: Mining Subspace Clusters in Categorical Data via *k*-Partite Maximal Cliques," *Proc. 21st Int'l Conf. Data Eng.,* 2005.
[34] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An Efficient Data Clustering Method for Very Large Database," *Proc. ACM SIGMOD,* 1996.
[35] A. Zhou, F. Cao, W. Qian, and C. Jin, "Tracking Clusters in Evolving Data Streams over Sliding Windows," *Knowledge and Information Systems,* 2007.

Hung-Leng Chen received the BS degree from National Taiwan Normal University, Taipei, in 2002. He is currently working toward the PhD degree in the Department of Electrical Engineering, National Taiwan University, Taipei. He majors in computer science and is familiar with the data mining area. His research interests include data mining and multimedia information retrieval. Among these, the Web mining, incremental mining, mining data streams, and categorical clustering are his special interests. He is a member of the IEEE.

Ming-Syan Chen received the BS degree in electrical engineering from National Taiwan University, Taipei, and the MS and PhD degrees in computer, information, and control engineering from the University of Michigan, Ann Arbor, in 1985 and 1988, respectively. He is now the director of research at the Center of Information Technology Innovation (CITI) and a distinguished research fellow at Academic Sinica, and also a distinguished professor in the Department of Electrical Engineering and the Graduate Institute of Communication Engineering, National Taiwan University. He was a research staff member at IBM Thomas J. Watson Research Center, Yorktown Heights, New York, from 1988 to 1996, the director of GICE from 2003 to 2006, and also the President/CEO of the Institute for Information Industry (III), which is one of the largest organizations for information technology in Taiwan, from 2007 to 2008. His research interests include database systems, data mining, mobile computing systems, and multimedia networking. He has published more than 270 papers in his research areas. In addition to serving as a program chair/vice chair and a keynote/tutorial speaker in many international conferences, he was an associate editor of the *IEEE Transactions on Knowledge and Data Engineering*, *Knowledge and Information Systems*, and the *Journal of Information Science and Engineering*, is currently on the editorial boards of the *VLDB Journal* and the *International Journal of Electrical Engineering*. He is a recipient of the National Science Council (NSC) Distinguished Research Award, Pan Wen Yuan Distinguished Research Award, Teco Award, Honorary Medal of Information, and K.-T. Li Research Breakthrough Award for his research work and also the Outstanding Innovation Award from IBM Corporate for his contribution to a major database product. He is a fellow of the ACM and the IEEE.

Su-Chen Lin received the BS degree in computer science and information engineering from the National Chiao Tung University, Hsinchu, Taiwan, in 2006. She is currently working toward the PhD degree in the Electrical Engineering Department, National Taiwan University, Taipei. Her research interest is data mining.