Extended Self-Organizing Map on Transactional Data

Wen-Chung Liao
Department of Information Management,
National Yunlin University of Science and Technology

Chung-Chian Hsu*
Department of Information Management,
National Yunlin University of Science and Technology

Abstract

In many application domains, transactions are the records of personal activities. Transactions always reveal personal behavior customs, so clustering the transactional data can divide individuals into different segments. Transactional data are often accompanied with a concept hierarchy, which defines the relevancy among all of the possible items in transactional data. However, most of clustering methods in transactional data ignore the existing of the concept hierarchy. Owing to the lack of the relevancy provided by the concept hierarchy, clustering algorithms tend to separate some similar patterns into different clusters. Besides, their clustering results are not easy to be viewed by users. The purpose of this study is to propose an extended SOM model which can handle transactional data accompanied with a concept hierarchy. The new SOM model is named as SetSOM. It can project the transactional data into a two-dimensional map; in the meanwhile, the topological order of the transactional data can be preserved and visualized in the 2-D map. Experiments on synthetic and real datasets were conducted, and the results demonstrated the SetSOM outperforms other SOM models in execution time, and the qualities of visualization, mapping and clustering.

Keywords: transactional data, self-organizing map, concept hierarchy, distance function on transactions, concept tree

* Corresponding author. Email: hsucc@yuntech.edu.tw
2011/03/19 received; 2011/08/17 revised; 2011/10/25 accepted

延伸自組映射圖探勘交易型資料

廖文忠
國立雲林科技大學資訊管理系

許中川*
國立雲林科技大學資訊管理系

摘要

在許多應用領域，交易紀錄反映個人行為上的偏好或習慣，若將交易紀錄適當分群，即可將不同行為類型的個人分到不同群組。交易型資料通常有概念階層伴隨，概念階層反映所有可能交易項目之間的相關性，然而，概念階層卻被大多數的分群演算法忽略，因此，易將相似度高的交易資料分屬不同群組；除此，分群結果通常不易被使用者觀看。本論文目的在延伸自組映射圖探勘具概念階層的交易資料，我們稱之為 SetSOM；SetSOM 可將交易資料映射至二維平面上，同時保有交易資料在其資料空間上的拓樸關係且可被觀看。利用人造資料及實際蒐集的交易型資料，進行實驗發現，SetSOM 無論在執行時間、視覺觀看品質、映射品質、及分群品質均高於其他自組映射圖的表現，包括 SCM 及 SOM。

關鍵詞：交易型資料，自組映射圖，概念階層，交易型資料距離函數，概念樹。

* 本文通訊作者。電子郵件信箱：hsucc@yuntech.edu.tw
2011/03/19 投稿；2011/08/17 修訂；2011/10/25

1. Introduction

Transactional data are common in many application domains such as marketing, finance, e-commerce, biology, medicine, etc. A typical example of transactional data is the market basket data, where each basket (or transaction) is a set of different items purchased by a customer in a transaction. Web pages browsed by visitors in a web site, books borrowed by students from their college library, and symptoms emerged in patients are another different examples. It is valuable to cluster transactional data into different groups. For example, the market basket data always reveal their customers' shopping behaviors and personal favorites. Clustering the market basket data can separate the customers into different segments, so managers of the shopping store can adopt different marketing strategies for the different segments of the customers to maximize their profits.

In many applications, transactional data are accompanied with a concept hierarchy or a taxonomy tree. Product catalogs of shopping stores, Web site structures, book categories of libraries etc. are examples of concept hierarchies. Concept hierarchies are tree structures (Han & Kamber 2001). All of the possible items in transactional data form the leaf nodes of the concept hierarchy. The non-leaf nodes are the categories (or concepts) of all the possible items. The relevancy among all of the possible items in transactional data is defined in the concept hierarchy. Items in the same category are more relevant than those of items in different ones. For example, in a shopping mall, apple juices are more relevant to grape juices than apples, since apple juices and grapes juices belong to the same juice category and apples are in the fruit category. If the apple juices are in shortage on the shelf, normally, customers tend to take a grape juice rather than an apple.

So far, many transactional data clustering methods had been developed and have well clustering quality (Guha et al. 2000; Hua et al. 2009; Wang et al. 1999; Yang et al. 2002). However, most of these clustering methods ignore the existing of concept hierarchies. The relevancy between individual items is never considered in most of the methods. Most of them put the transactions which have more items in common into the same cluster. The important information between the items is lost. It may cause that the transactions whose items are different but relevant are considered as different kinds of patterns. For example, if we have two transactions t_1={apple juice, orange, coke} and t_2={pesi, grape juice, apple} that come from a shopping mall. Most of the clustering

methods, like LargeItems (Wang et al. 1999), Clope (Yang et al. 2002), Scale (Hua et al. 2009), would not put these two transactions into the same cluster, since these two transactions do not have any items in common. Put them into the same cluster would reduce the number of large items (Wang et al. 1999), slope (Yang et al. 2002), or density (Hua et al. 2009) of the cluster. It is easy to understand that these two transactions are similar since their items are highly relevant.

Moreover, most of the clustering methods have difficulty in visualizing their clustering results, so the structure of the clusters can not be inspected by users easily. The self-organizing map (SOM), proposed by Kohonen (1982, 1995), has high quality in projecting high dimensional numerical data into a low dimensional space, typically a two-dimensional map, such that the projecting result can be viewed. Thanks to its well visualization property, the SOM has been applied to many related applications (Kohonen et al. 1996, 2000; Yeh & Chang 2006), including data clustering analysis (Chen et al. 2000).

Recently, some SOM models like the SCM (Flanagan 2003; Himberg et al. 2003), the TCSOM (He et al. 2005) had been developed to handle transactional data, but they do not consider the existing of concepts hierarchies too. On another side, some SOM models, like the GSOM (Hsu 2006) and the GViSOM (Hsu et al. 2006) were proposed to handle mixed-type data. Although they take concept hierarchies, which are associated with each categorical feature, into account, they can not apply to transactional data well since their data objects are thought as fixed dimensional vectors, not sets.

In this paper, an extended SOM model which can handle set-type transactional data accompanied with a concept hierarchy is proposed. We named it as SetSOM. SetSOM can project the transactional data into a two-dimensional map; in the meanwhile, the topological order of the transactional data can be preserved and visualized in the 2-D map.

This study has the following contributions. The first one is the SetSOM extends the application scope of the conventional SOM to manage the transactional data with a concept hierarchy. The second is a distance function is given to measure the distances between transactions such that the relevancy of the items takes into consideration. The third is the data structure of the transactional data with a concept hierarchy can be precisely inspected by a visualization map.

The rest of this paper is organized as follows. In section 2, the SOM and some extended models are reviewed. Section 3 gives the methodology of the SetSOM. Experiments are reported in section 4. Conclusions and future works are described in

section 5.

2. Literature review

2.1 Self-organizing map

The self-organizing map (SOM) is an unsupervised neural network (Kohonen 1982, 2001). It can project high dimensional data objects into a lattice of neurons, typically arranged on a 2-D map; in the meanwhile, the topological order among the data objects in their data space can be preserved on the 2-D map. The lattice can be rectangular or hexagonal. Each neuron has its own reference vector, which belongs to the same space of the data objects. After learning from input patterns, the reference vectors of the neurons will reflect the distribution of the data objects in their own data space.

The neurons learn from the input patterns randomly and iteratively. In the traditional SOM, there are two key steps whenever an input pattern is randomly drawn from a dataset. The first one is to find the best matching unit (BMU) from all of the neurons on the map. If x is an input pattern, then the best matching unit c is found as follows.

$$c = \arg\min_i \| x - m_i \|, \tag{1}$$

where m_i is the reference vector of the neuron i. That is, among all the neurons, c is the one which has minimal distance between its reference vector and x.

The second key step is to adjust the reference vectors of the neurons which are in the neighborhood of the BMU. Let i be a neuron which is in the neighborhood of c. The reference vector m_i of the neuron i is adjusted by the following equation.

$$m_i(s+1) = m_i(s) + h_{ci}(s)[x(s) - m_i(s)], \tag{2}$$

where $h_{ci}(s)$ is the neighborhood function, $0<h_{ci}(s)<1$, and s denotes the current time. The Gaussian kernel is widely used in the neighborhood function $h_{ci}(s)$ as follows.

$$h_{ci}(s) = \alpha(s) \cdot \exp\left(-\frac{\| r_c - r_i \|^2}{2\sigma^2(s)} \right), \tag{3}$$

where $\alpha(s)$ is the learning rate, $0<\alpha(s)<1$, $\sigma(s)$ is the width of the Gaussian kernel, and r_c and r_i are the locations of the neurons c and i on the map, respectively. Both $\alpha(s)$ and $\sigma(s)$ are monotonically decreasing with time s. From equation (2), we have

$$m_i(s+1) = [1 - h_{ci}(s)]m_i(s) + h_{ci}(s)x(s) \tag{4}$$

That is, the newly reference vector of the neuron i is a linear combination of $m_i(s)$ and $x(s)$. Since $[1\text{-}h_{ci}(s)]+ h_{ci}(s)=1$, $m_i(s+1)$ will be the point on the line segment between $m_i(s)$ and $x(s)$ as depicted in Fig. 1.

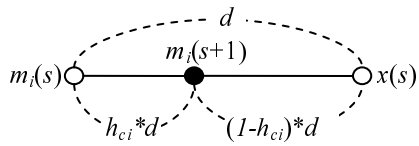


Figure 1：Adaptation of reference vector

Let $d = |x(s) - m_i(s)|$. Then the reference vector of the neuron i after adaptation, i.e., $m_i(s+1)$, must be the vector that satisfies the following two equations:

$$|m_i(s+1) - m_i(s)| = h_{ci}(s) \cdot d \ , \tag{5}$$

$$|m_i(s+1) - x(s)| = [1 - h_{ci}(s)] \cdot d \ . \tag{6}$$

That is, after adaptation, the reference vector m_i is more similar and closer to x.

2.2 SOM on non-vector data

The SOM is suitable for the n-dimensional vector dataset. But Kohonen (1996) had shown that any non-vector dataset with a defined distance measure or similarity can be mapped by the SOM. It is based on the use of batch-SOM training and the median associated with each neuron (Kohonen & Somervuo 1998, 2002). Afterward the online SOM algorithm for non-vector data such as symbol strings (Somervuo 2004), graphs (Günter & Bunke 2002), transactions (Flanagan 2003; Himberg et al. 2003; He et al. 2005), time series (Hammer et al. 2005), and so on had been developed.

The Symbol String Clustering Map (SCM) is an unsupervised clustering method for symbol string data (Flanagan 2003; Himberg et al. 2003). Symbol strings are composed of symbols. Like the SOM, the SCM has a lattice of nodes on a 2-D map. Each node of the SCM is associated with a symbol string and a weight vector. Each

symbol of the node string is associated with one coefficient in the weight vector. Whenever an input symbol string is drawn from the dataset and fed to the SCM, the winner node, which has the maximum activation value, is found. Next, an adaptation step is followed. For each node, both of its node string and weight vector are adjusted based on the input symbol string, the winner node, the values of the learning rate and the kernel function at time t. As a result, the non-null nodes form the clusters and are separated by the null nodes on the map. Transactions are symbol strings if their items are considered as symbols, but SCM does not consider the concept hierarchy.

In the paper of Günter and Bunke (2002), graphs can be clustered by the SOM. Distance between two graph g_1 and g_2 is defined as the edit distance, which is the minimal cost when a sequence of edit operations including substitution, deletion, and insertion are applied to g_1 such that the edited graph g_1 is isomorphic to the graph g_2. Moreover, a weighted mean of g_1 and g_2 is defined as a graph g such that, for some α with $0 \leq \alpha \leq d(g_1, g_2)$, the following two equations hold:

$$d(g_1, g) = \alpha, \tag{7}$$

$$d(g_1, g_2) = \alpha + d(g, g_2). \tag{8}$$

In this SOM model, graphs are directly used as the neuron's model. Thus, whenever an input pattern, a graph x, chosen from the dataset, the BMU can be found among all of the neurons by the edit distance measure; and the graphs of the neurons around the neighborhood of the BMU can be adjusted via the above concept of the weighted mean. If y is the graph of a neuron in the neighborhood of the BMU and γ is the value of the neighborhood function, then the new graph of y after adjusting is determined by the following equations:

$$d(y, y_{\text{new}}) = \gamma d(x, y), \tag{9}$$

$$d(x, y) = d(x, y_{\text{new}}) + d(y_{\text{new}}, y). \tag{10}$$

That is, the graph y_{new} is the weighted mean of x and y. These two equations follow the equations (7) and (8) directly, and they are the same as the equations (5) and (6) in the SOM. This SOM model is used to cluster characters.

A transaction can be considered as an 1-level tree if all its items are treated as the leaf nodes. Thus the above SOM model can be applied to transactional data. However, bias could occur when the non-normalized edit distance is applied to transactions.

3. Methodology

In this section, an extended SOM model which can handle transactional data with a concept hierarchy is proposed. We call it SetSOM. Before the proposed SetSOM, a distance function defined on transactions and three tree operators are given.

3.1 Preliminaries

Let $I=\{i_1, i_2, \ldots, i_M\}$ be a set of M possible items. Let D be a transactional dataset. That is, D is a collection of transactions. We assume the size of D is N. Suppose t is a transaction of D. Then t is a set of distinct items of I, that is, t is a subset of I.

Let H be a concept hierarchy of I. It means that H must be a rooted and labeled tree with all items of I as its leaves. The non-leaf nodes of H act as taxonomic roles of the items in I. The top-level non-leaf nodes are the main categories of I, and the higher level ones are those of subcategories of I. Hence we call the leaves in H as **item nodes** and the non-leaf nodes as **category nodes**. For convenience sake, the root node of H is labeled as r, and every node in H is considered to have a unique label. Moreover, the left-to-right order among the siblings of any category node in H is significant. A three levels concept hierarchy is depicted in Fig. 2.

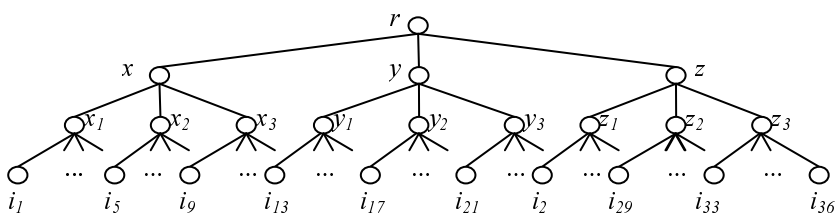


Figure 2：A three levels concept hierarchy. The nodes x, y, and z are the main categories; the nodes x_is, y_is, and z_is are the subcategories; the i_js are the items.

Let T be a subgraph of H. All the nodes and the edges in T must be in H. If T is a tree with the same root of H, T is called a **concept tree** which comes from H. T is ordered and labeled too. If a is a node of T, a value is given and called the growth value of a, denoted by $g_T(a)$. The growth value is greater than zero and not greater than one. When $g_T(a)=1$, a is said to be fully grown in T. When $0<g_T(a)<1$, a is partially grown in T. In any concept tree T, only fully grown category nodes can have their child

nodes. If a is a category node in T, the child nodes of a in T must be also child nodes of a in H. If T contains item nodes of H, these nodes must be leaves of T. The weight of T, denoted by $|T|$, is the total growth values of all the nodes in T, that is, $|T| = \Sigma_{a \in T} g_T(a)$. Based on the concept hierarchy in Fig. 2, an example of concept tree is shown in Fig. 3(a). The nodes x, y, and y_2 are fully grown, and the nodes x_1 and i_{17} are partially grown. If all the item nodes of T are fully grown and exactly equal to those of the items in transaction t, we call T the corresponding **transaction tree** of t. If we have a transaction $t=\{i_1, i_4, i_{17}, i_{21}\}$, then a transaction tree T with $|T|=10$, which corresponds to t, is depicted in Fig. 3(b).

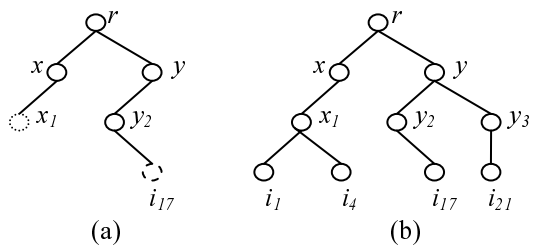


Figure 3 : Examples of a concept tree and a transaction tree. (a) is a concept tree; (b) is a transaction tree T corresponding to a transaction $t=\{i_1, i_4, i_{17}, i_{21}\}$ with $|T|=10$.

A concept tree, like the example in Fig. 3(a), can be represented as a class of transactions. Any transaction which has the item i_{17} and the items belonging to the subcategories x_1 or y_2 has a high possibility in the class of this concept tree. Besides, the transactions in the above class have a higher possibility in containing the items which belong to the subcategory y_2 and a less possibility in containing the items which belong to the subcategory x_1, since y_2 is more fully grown than x_1 in this concept tree. Concept trees will be used as the prototypes of neurons in our SOM model.

3.2 Distance between transactions

Before we have a distance function defined on transactions, a distance function defined on concept trees is first given. Here are some notations. For any tree T, $\Psi_T(a|b)$ is denoted as a subtree rooted at the node a with b as its parent node, if no ambiguity, b can be ignored, and $\pi_T(b)$ is the set of all the child nodes of b. Based on the Jaccard's coefficient (Lampinen and Oja 1992), a distance function of any two trees, T and U, with a common root node r is given as follows.

$$\text{dist}(T,\ U) = \frac{\text{Wdiff}(T,\ U)}{\text{Wintersect}(T,\ U) + \text{Wdiff}(T,\ U)} \tag{11}$$

where

$$\text{Wdiff}(T,\ U) = \sum_{a \in \pi_T(r) - \pi_U(r)} |\Psi_T(a \mid r)| + \sum_{a \in \pi_U(r) - \pi_T(r)} |\Psi_U(a \mid r)| + \sum_{a \in \pi_T(r) \cap \pi_U(r)} d(\Psi_T(a \mid r), \Psi_U(a \mid r)), \tag{12}$$

$$\text{Wintersect}(T,\ U) = \sum_{a \in \pi_T(r) \cap \pi_U(r)} \frac{|\Psi_T(a \mid r)| + |\Psi_U(a \mid r)| - d(\Psi_T(a \mid r), \Psi_U(a \mid r))}{2}, \tag{13}$$

and

$$d(\Psi_T(a \mid r), \Psi_U(a \mid r)) = \begin{cases} \left| |\Psi_T(a \mid r)| - |\Psi_U(a \mid r)| \right| & \text{either } \Psi_T(a \mid r) \text{ or } \Psi_U(a \mid r) \text{ degenerates to node } a, \\ \text{dist}(\Psi_T(a \mid r), \Psi_U(a \mid r)) & \text{otherwise.} \end{cases} \tag{14}$$

Note that Wdiff(T, U) and Wintersect(T, U) are used to measure the mutual differences and the intersection between T and U, respectively. We explain these two functions in the next example. The function $d(\Psi_T(a|r), \Psi_U(a|r))$ is used to calculate the difference between the two subtrees $\Psi_T(a|r)$ and $\Psi_U(a|r)$. If either $\Psi_T(a|r)$ or $\Psi_U(a|rb)$ degenerates to a node a, we directly calculate their weight difference as their difference. Otherwise, we recursively use the dist(·,·) to calculate the distance between $\Psi_T(a|r)$ and $\Psi_U(a|r)$ as their difference. In the latter case, we tend to treat $\Psi_T(a|r)$ and $\Psi_U(a|r)$ are similar, since both of them are concept trees with a common fully grown root node a, and can be considered as two groups of relevant items under the same category a. Thus, we use distance function to measure their difference instead of using their weight difference. Usually, the weight difference between $\Psi_T(a|r)$ and $\Psi_U(a|r)$ are much bigger than the distance.

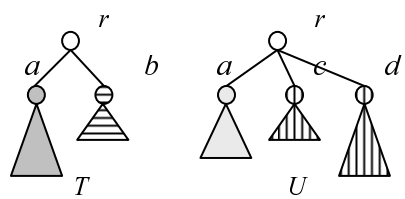


Figure 4：Two concept trees and their first level subtrees.

We further explain the distance function by an example in Fig. 4. From the point of view of the two root nodes, the subtree $\Psi_T(b|r)$ is completely in T and not in U, vice versa the subtrees $\Psi_U(c|r)$ and $\Psi_U(d|r)$ are completely in U and not in T. $\Psi_T(a|r)$ and $\Psi_U(a|r)$ are two similar subtrees because all of the items in both of the two subtrees belong to the same category node a. However, differences may occur inside the two subtrees. Thus we recursively use the distance function to measure their differences. Therefore, by the definition of Wdiff(·,·), the total mutual differences between T and U are as follows:

$$\text{Wdiff}(T, U)=|\Psi_T(b|\ r)|+ |\Psi_U(c|r)|+|\Psi_U(d|r)|+\text{dist}(\Psi_T(a|r), \Psi_U(a|r)) \tag{15}$$

Again, from the two root nodes' view, $\Psi_T(a|r)$ and $\Psi_U(a|r)$ are in common once their difference has been subtracted. So, by the definition of Wintersect(·,·), the intersection between T and U is calculated by the following average:

$$\text{Wintersec}(T, U) = \frac{|\Psi_T(a|r)| + |\Psi_U(a|r)| - \text{dist}(\Psi_T(a|r), \Psi_U(a|r))}{2} \tag{16}$$

Such a distance function defined on concept trees can measure the differences in their tree structures. Now, based on the above distance function defined on concept trees, we give a distance measure between any two transactions t_i and t_j as follows.

$$\text{dist}(t_i, t_j)=\text{dist}(T_i, T_j) \tag{17}$$

where T_i and T_j are the corresponding transaction trees of t_i and t_j, respectively. It measures not only the differences between transactions in their items but also the differences between transactions in their trees structures. That is, based on the concept hierarchy H, the distance measure takes the relevancy between items into account.

We demonstrate the characteristics of the distance function by the following example. Suppose there are four transactions t_1={a}, t_2={b}, t_3={c}, and t_4={a, c} whose items come from some concept hierarchy, and their corresponding transaction trees are depicted in Fig. 5, respectively. Then, we have dist(t_1, t_2)=1/8, dist(t_1, t_3)=1, dist(t_2, t_3)=1, and dist(t_1, t_4)=1/2 by using the above distance function. If the traditional Jaccard's coefficient is applied, then the relationship of the items in the concept hierarchy is ignored, and we have dist(t_1, t_2)=1, dist(t_1, t_3)=1, dist(t_2, t_3)=1, and dist(t_1, t_4)=1/2. The similarity among t_1, t_2, t_3, and t_4 cannot be identified.

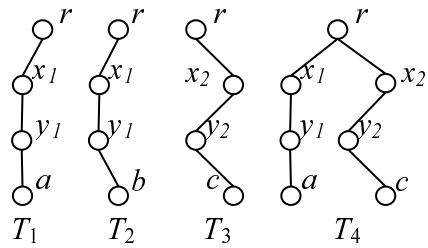


Figure 5：Transaction trees of t1, t2, t3 , and t4.

3.3 Operators on concept trees

In order to modify the concept trees in our SetSOM, we extend three set operators, union, intersection and difference, on concept trees. Let T and U be two concept trees of H. $T \cap U$ and $T \cup U$ are denoted as the intersection and the union of T and U, respectively. T-U is denoted as the difference of T from U. Firstly, $T \cap U$ is defined as the concept tree with nodes in both T and U. Let a be a node in $T \cap U$. We define $g_{T \cap U}(a) = \min(g_T(a), g_U(a))$. Next, $T \cup U$ is defined as the concept tree with nodes in T or U. Let b be a node in $T \cup U$. The growth value of b is determined as follows.

$$g_{T \cup U}(b) = \begin{cases} g_T(b) & \text{if } b \in T \text{ and } b \notin U \\ g_U(b) & \text{if } b \notin T \text{ and } b \in U \\ \max(g_T(b),\ g_U(b)) & \text{if } b \in T \text{ and } b \in U \end{cases} \tag{18}$$

The tree difference of T from U, T-U, is defined as follows. If a node c is in T and not in U, then c is in T-U, and its growth value is $g_T(c)$. If c is in both of T and U, and $g_T(c) > g_U(c)$, then c is in T-U, and its growth value is $g_T(c)$-$g_U(c)$. Otherwise, c is not in T-U. Thus, if c is a node in T-U, the growth value of c is determined as follows.

$$g_{T-U}(c) = \begin{cases} g_{T_i}(c) & \text{if } c \in T \text{ and } c \notin U \\ g_{T_i}(c) - g_U(c) & \text{if } c \in T, c \in U, \text{and } g_T(c) > g_U(c) \end{cases} \tag{19}$$

The result of T-U is no more a concept tree. It is possible that T-U could contain many subtrees of T. For example, if U is composed of only one root node r with fully grown, and T is a concept tree whose root node r has three fully grown child nodes, say,

w_1, w_2, and w_3, then T-U would contain three subtrees of T whose root nodes are those three child nodes, w_1, w_2, and w_3, of r in T. They are no longer concept trees.

Let T and U be any two concept trees of H, then we have the following properties: (i) $|T \cup U| = |U - T| + |T - U| + |T \cap U|$ and (ii) $|T \cup U| = |T| + |U - T|$. Follow the definitions of the above three operators, the two properties can be proved.

3.4 SetSOM and its algorithms

3.4.1 Initialization phase

The SetSOM is a self-organizing map whose neurons are arranged in a lattice on a two-dimensional map. The lattice can be rectangular or hexagonal. Instead of using reference vectors as the prototypes in the conventional SOM, the prototype of each neuron in the SetSOM is a concept tree of the concept hierarchy H. It is quite different from the other SOM models at this point. Thus, before learning from input patterns, each neuron in the SetSOM will be initially assigned a concept tree as its prototype. Those concept trees can be randomly generated based on the concept hierarchy H. For convenient sake, for each neuron, a transaction is randomly drawn from the dataset then its corresponding transaction tree is simply assigned as the neuron's prototype.

3.4.2 Training phase

In this phase, the SetSOM learns from input patterns randomly and iteratively. The input patterns are the transactions from the dataset D. Once a transaction is randomly drawn from D, the SetSOM learns from it. The learning process is continued until some criterion is met. In our later experiments, criterions are usually predefined and fixed iteration numbers, but it is related to the number of the neurons on the map (Kohonen et al. 1996; Vesanto et al. 2000).

The same as the traditional SOM models, there are two key steps whenever a transaction is given in this training phase. The first one is to find the best matching unit among all the neurons on the map. The second is to adapt the prototypes of the neurons which are in the neighborhood of the BMU. In the first step, suppose t is the transaction randomly chosen from the dataset D, and its corresponding transaction tree T is fed to the SetSOM. Then the BMU c can be found as follows.

$$c = \arg\min_i dist(T, U_i), \tag{20}$$

where U_i is the concept tree of the ith neuron in the SetSOM. That is, the BMU is the

neuron which has the smallest distance between its concept tree and the given transaction tree.

In the second step, the neurons in the neighborhood of the BMU learn from the transaction *t*. The transaction tree of *t* is denoted as $T(s)$ to emphasize the current time is at *s*. Those concept trees of the neurons in the neighborhood of the BMU are adjusted according to $T(s)$ as well as the neighborhood function h(s).

Let *i* be a neuron in the neighborhood of the BMU and $U_i(s)$ be the concept tree of the neuron *i* at the time *s*. According to the adaptation mechanism of the conventional SOM, which we have described in our previous section, $U_i(s)$ should be adjusted toward $T(s)$ such that the distance between them is decreased. That is, if $dist(T(s), U_i(s)) = d$, the concept tree of the neuron *i* after modification, i.e., $U_i(s+1)$, must satisfies the following two equations:

$$dist(T(s), U_i(s+1)) = d \cdot (1 - h_{ci}(s)) \tag{21}$$

$$dist(U_i(s), U_i(s+1)) = d \cdot h_{ci}(s) \tag{22}$$

where $h_{ci}(s)$ is the value of the neighborhood function at the time *s*. That is, $d \cdot h_{ci}(s)$ should be the total adjustment from $U_i(s)$ to $U_i(s+1)$ such that $U_i(s+1)$ is more similar and closer to the input transaction tree $T(s)$. Nevertheless, It is difficult to have $U_i(s+1)$ which satisfy both of the equations (21) and (22) especially when treating non-vector data. The similar case happens in the study of Günter and Bunke (2002). It can be proved that $U_i(s+1)$ which satisfy both of the equations (21) and (22) is impossible, when the distance function is defined below. Owing to the limit in pages, we omit the proof.

$$\text{dist}(T,\ U) = \frac{|T - U| + |U - T|}{|T \cup U|} \tag{23}$$

Next, we describe the way how we adjust the concept tree $U_i(s)$ of the neuron *i* such that $U_i(s+1)$ can get more similar to $T(s)$ at time s. For convenient sake, the concept trees of $U_i(s)$ and $T(s)$ are simplified to Fig. 6(a) and 6(c), respectively. The areas filled with little dots in Fig. 6(a) and 6(c) are the common parts of $U_i(s)$ and $T(s)$. The area filled with horizontal lines in Fig. 6(a) represents the tree difference of $U_i(s)$ from $T(s)$, i.e., $U_i(s) - T(s)$. We call it the negative part of $U_i(s)$, since we expect a fraction of this part will be pruned from $U_i(s)$. The area filled with vertical lines in Fig.

6(c) represents the tree difference of $T(s)$ from $U_i(s)$, i.e., $T(s)-U_i(s)$. We call it the positive part of $U_i(s)$ at this adaptation, since we expect a fraction of this part will grow up in $U_i(s)$.

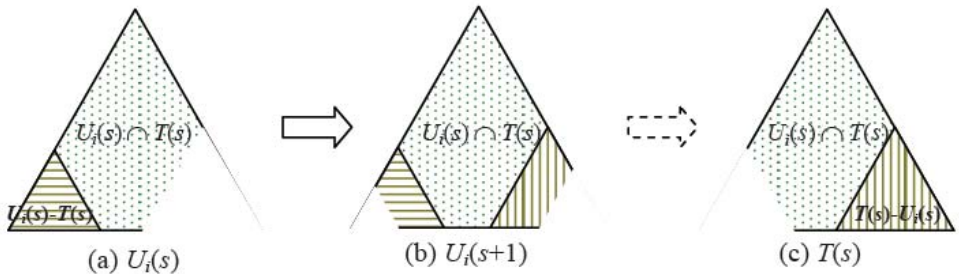


Figure 6：Adaptation of the concept tree $U_i(s)$ of neuron i when $T(s)$ is given at time s.

Therefore, a reasonable way to adjust the concept tree $U_i(s)$ of the neuron i such that $U_i(s)$ can get more similar and closer to $T(s)$ is to prune a fraction of the negative part from $U_i(s)$ and to grow a fraction of the positive part in $U_i(s)$. Suppose the fraction which will grow up in $U_i(s)$ takes a percentage h_1 of the positive part and the fraction which will be pruned from $U_i(s)$ takes a percentage h_2 of the negative part. Fig. 6(b) represents the concept tree of the neuron i after adaptation.

Let $W_{pos}=|T(s)-U_i(s)|$ and $W_{neg}=|U_i(s)-T(s)|$. Once the percentages h_1 and h_2 are determined, $W_{\text{pos}}\times h_1$ of the weight of the positive part will grow in $U_i(s)$ and $W_{neg}\times h_2$ of the weight of the negative part will be pruned from $U_i(s)$. We call $W_{pos}\times h_1$ and $W_{neg}\times h_2$ the positive and negative adjustment value, respectively.

Within the positive part, there may contain many subtrees of $T(s)$ with different weights. If Y is one of the subtrees with weight $|Y|$, then $W_{pos}=\sum_{Y\in T(s)-U_i(s)}|Y|$. The leaves or nodes starting from the root node of Y will grow up in $U_i(s)$. It is possible the root node of Y has partially grown in $U_i(s)$. Totally a fractional weight, $|Y|\times h_1$, of Y will grow in $U_i(s)$. Moreover, the parent node of the root node of Y is called the growing point of Y in $U_i(s)$.

Similarly, within the negative part, there may contain many subtrees of $U_i(s)$ with different weights. If Y is one of the subtrees with weight $|Y|$, then $W_{neg}=\sum_{Y\in U_i(s)-T(s)}|Y|$. The leaves or nodes of Y will be pruned from U. Totally, a fractional weight, $|Y|\times h_2$, of Y will be pruned from $U_i(s)$.

In Fig. 7, we illustrate the subtrees in the negative and the positive parts of a

concept tree $U_i(s)$ when an input pattern $T(s)$ is given. Suppose x_1 is partially grown in $U_i(s)$, and then the growing points are the nodes r and w_1 in $U_i(s)$, from where the subtrees rooted at w_2, w_3, x_1, and x_2 will grow.

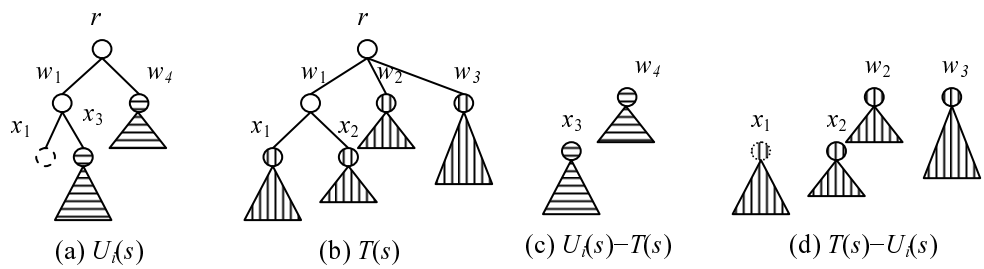


Figure 7：The subtrees in the negative and the positive part of a concept tree Ui(s) when an input pattern T(s) is given. (a) Ui(s) and the growing points, r and w1. (b) T(s). (c) The subtrees in the negative part. (d) The subtrees in the positive part. Since the node x1 in Ui(s) is partial grown, x1 is only partially in T(s)-Ui(s).

So far, based on the adaptation mechanism we have described, the adapted concept tree gets similar and closer to the input pattern. However, It is difficult to have $U_i(s+1)$ which satisfy both of the equations (21) and (22). Thus, we take $h_1=h_2=h_{ci}(s)$ heuristically and simply. Although the incurable adaptation error could exist, $U_i(s+1)$ gets far away from $U_i(s)$, and $U_i(s+1)$ gets closer to $T(s)$.

3.4.3 Adaptation algorithms of SetSOM

Now the entire algorithm of the adaptation step is shown in Fig. 8. Just as we have mentioned above, a transaction tree $T(s)$ and the BMU at time s are given as the inputs. At the beginning of the algorithm, the learning rate $\alpha(s)$ and the updating radius $\sigma(s)$ are recalculated at this step s; and the location of BMU is determined on the map.

Then for each neuron i in the neighborhood of BMU, its prototype, i.e., the concept tree $U_i(s)$, of the neuron is modified based on the input pattern $T(s)$ as well as the value of the neighborhood function $h_{ci}(s)$. During the modification process, the positive and the negative parts are determining as well as the growing points. It is easy to find out the subtrees in both of the positive and the negative parts of $U_i(s)$, when we traverse the trees nodes of $U_i(s)$ and $T(s)$ from their roots and recursively compare their child nodes.

```
PrototypeAdaptation(T, BMU, s)
    Input: Input pattern T, BMU, training time s
    Output: Updated prototypes of the neighboring neurons around BMU
    Recalculate α(s) and σ(s) such that they are monotonically decreasing with s;
    Let (x,y) be the location of the BMU in the map;
    Let N(x, y) be the neighborhood centered at (x, y) within a specified range, σ(s);
    For each neuron i in N(x, y)
        Let positive links the subtrees in T- U_i
        Let growPoints links the nodes in U_i that new branches will grow
        Let negative links the subtrees in U_i -T
  Let (u,v) be the location of i in the map;
 Let r= dist( (u, v), (x, y) );              // Euclidean distance between i and BMU on the Map
  Let h=α*exp(-(r*r)/(2*σ*σ) );               // the neighborhood function
  Let h_1 = h_2= h;
  U_i.ManySubtreesGrow( growingPoints, positive, h_1);
  U_i.ManySubtreesPrune( negative, h_2);
```

Figure 8：Algorithm of prototype adaptation.

Then by using the ManySubtreesGorw algorithm, which is depicted in Fig. 9, a fraction of the positive part grows up in U_i. Each subtree in the positive part will grow up under the OneSubtreeGrow algorithm, which is similar to the ManySubtreesGorw algorithm, so we omit it. Similarly, by using the ManySubtreesPrune algorithm, also depicted in Fig. 9, a fraction of the negative part is pruned from U_i. Each subtree in the negative part is pruned by the OneSubtreePrune algorithm, we omit it, too.

```
ManySubtreesGrow(growingPoints, positive, h_1)
  x=growingPoints;
  y=positive;
  while(y!=null)
      g=y.weight * h_1;
      OneSubtreeGrow(x.root, y.root, g);
      y=y.next;    // next subtree in positive link
      x=x.next;    // next growing point in U
```

```
ManySubtreesPrune(negative, h_2)
  y=negative;
  while(y!=null)
      p=y.weight * h_2;
      OneSubtreePrune(y.root, p);
      y=y.next;    // next subtree in negative link
```

Figure 9：Algorithm of ManySubtreesGrow and ManySubtreesPrune.

3.5 Evaluation of mapping and clustering quality

3.5.1 Mapping quality

Two measures are used to observe the mapping qualities: the trustworthiness and the continuity functions. When the nearest neighbors of an object in the mapped space are also close in the original data space, it is said the mapping is trustworthy (Vathy-Fogarassy & Abonyi 2009; Kaski et al. 2003; Venna & Kaski 2005). Let N be the number of the objects to be mapped, $U_k(i)$ be the set of objects that are the k nearest neighbors of the data object i in the visualization map, but not in the original data space. The trustworthiness of the mapping can be calculated as follows.

$$\text{Trustworthness}(k) = 1 - \frac{2}{Nk(2N - 3k - 1)} \sum_{i=1}^{N} \sum_{j \in U_k(i)} (r(i, j) - k) \tag{24}$$

where $r(i,j)$ denotes the rank of the object j relative to i in data space.

When objects near to each other in the data space are also close on the map, it is said the mapping is continuous (Vathy-Fogarassy & Abonyi 2009; Kaski et al. 2003; Venna & Kaski 2005). The measure of the continuity of a mapping is calculated as follows.

$$\text{Continuity}(k) = 1 - \frac{2}{Nk(2N - 3k - 1)} \sum_{i=1}^{N} \sum_{j \in V_k(i)} (s(i, j) - k) \tag{25}$$

where $s(i,j)$ is the rank of the data object j relative to i in the output space, and $V_i(k)$ denotes the set of those data objects that are the k nearest neighbors of data object i in the original space, but not in the mapped space. The larger values of both of the trustworthiness and the continuity are better in mapping quality.

3.5.2 Clustering quality

We evaluated the clustering quality by the F measure and the weighted entropy. The F measure is defined as follows.

$$F = \sum_{i} \frac{N_i}{N} \max_{j} F(i, j) \tag{26}$$

where $F(i,j)=2P(i,j)R(i,j)/(P(i,j)+R(i,j))$, $P(i,j)=N_{ij}/N_j$, and $R(i,j)=N_{ij}/Ni$, N_i is the number of objects belonging to class i, N_j is the number of objects belonging to cluster j, N_{ij} is

the number of objects belonging to class i in cluster j. N is the total number of objects in dataset, $P(i, j)$ is the precision of cluster j in class i, and $R(i, j)$ is the recall of class i in cluster j. $F(i, j)$ is the F measure of cluster j in class i. The larger F is better in clustering quality. The second measure of clustering quality is the weighted entropy defined as follows.

$$\text{Weighted Entropy} = \sum_j \left(\frac{N_j}{N} \sum_i \left(-\frac{N_{ij}}{N_j} \cdot \log \frac{N_{ij}}{N_j} \right) \right) \tag{27}$$

The smaller weighted entropy is better in clustering quality.

4. Experiments

We conducted four experiments to evaluate the different characteristics of the SetSOM. Synthetic datasets were used in the first three experiments to demonstrate the projecting ability of the SetSOM and its mapping quality. The first experiment is a robustness testing of the SetSOM. The second and the third ones are to demonstrate the properties of the topological preservation and the visualization effect of the SetSOM, besides, the mapping quality is observed. By using one real dataset, the fourth experiment was conducted to investigate the SetSOM in clustering performance preliminarily. Comparisons with the SOM and the SCM are performed in all of the experiments.

The neurons of the three comparing models, SetSOM, SOM, and SCM, are all arranged in rectangular lattices in their own 2-D maps. The parameters setting of both the SetSOM and the SOM refer to the suggestions of the SOM_PAK (Kohonen et al. 1996) and the SOM_Toolbox (Vesanto et al. 2000). The number of the neurons in a map is about $5\sqrt{N}$, where N is the number of data objects in a dataset.

In the SetSOM and the SOM, the Gaussian function is used in the neighborhood function. The learning rate is of the form, $\alpha(s)=\alpha(0)\times(1.0–s/S)$, where the initial value $\alpha(0)=0.5$, s is the time, and S is the total iteration times. The updating radius is of the form, $\sigma(s)=1+(\sigma(0)–1)\times(1.0–s/S)$ with $\sigma(0)$ is about half of the width of the map. The number of training iterations is set to 10 times of the number of neurons. In the SCM, the Mexican hat is used as its neighborhood function according the author's suggestion (Flanagan 2003). The Mexican hat is defined as follows.

$$h(i) = (1 - a \cdot b \cdot i^2) \cdot \exp(-a \cdot i^2) \tag{28}$$

where a=0.35 is a constant, determining the width of Mexican hat and b=2.0×s/S varies with time s. The learning rate is $\alpha(s)$=0.25×S/(0. 5×S +s), and the value of the threshold is $x(s)$=0.2×s/S

Concept hierarchies are used for the SetSOM in all of the following experiments, of course. In the SOM, only the item nodes in concept hierarchies are used, and transactions are needed to convert to binary vectors. For example, if a transaction has only the 5th, 9th, and 11th items which come from a concept hierarchy with 12 item nodes, then its corresponding binary vector is (0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0). Anyway, both the SOM and the SCM ignore the information in concept hierarchies.

4.1 Robustness testing experiment

Since the amount of transactions is very tremendous in many applications, in this experiment, we test the robustness of the SetSOM. Before doing this experiment, a virtual retailer which has 100,000 products (items) is created. Suppose all of the products form a three levels concept hierarchy as follows. There are 20 main categories, 50 subcategories per each main category, and 100 products per each subcategory.

Three datasets, which have 10k, 50k, and 100k transactions, are created, respectively. All of the transactions are simulated by the following rules. Let T be the size of any transaction and X be a Poisson distribution with $E(X)$=14. Assume T=X+1. That is, once X is given, T is given. Then, for each transaction, there are T items drawn randomly from the 100,000 items without replacement Besides, $E(T)$=$E(X)$+1=15, that is, there are averagely 15 items for each transaction in this retailer.

For each dataset, the three comparing models are executed respectively, and their execution times are recorded. In Fig. 10, the dataset with 10k transactions is mapped by the SetSOM. Each neuron with nonzero BMU count is displayed by a circle. The area of the circle is proportional to its own BMU count. The 10k transactions seem uniformly distribute on the 2-D map. This is what we expect. Besides, we compare the execution time of the three comparing models in the training phase with the three different datasets; the result is depicted in Fig. 11. It is obvious that the SetSOM outperforms both the SCM and the SOM.

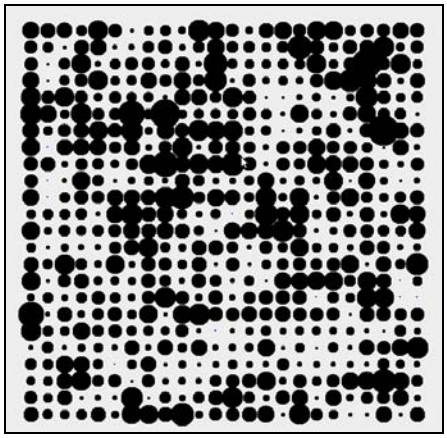


Figure 10：10k transactions of virtual retailer mapped by SetSOM on a 24x24 units.

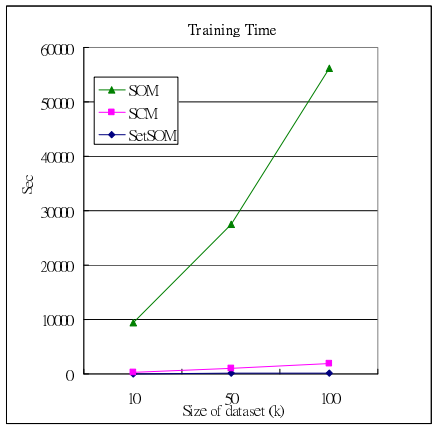


Figure 11：Comparison of SetSOM, SCM and SOM in training time for 10k, 50k and 100k transactions

4.2 Simulative library experiment

In the following experiment, we observe the visualization and the mapping qualities of the SetSOM and compare with the other two models. A simulative college library which has 1,250 books is created. Suppose all of the books form a three levels concept hierarchy as follows. (1) Level-1 nodes: there are five main categories (level-1 nodes) which are computer science (CS), mathematics (MA), physics (PH), biology (BI), and chemistry (CH). (2) Level-2 nodes: each main category has five subcategories (level-2 nodes), e.g., PH.1, PH.2, PH.3, PH.4, PH.5 are the five subcategories of the main category PH. (3) Level-3 nodes (leaves): each subcategory has 50 books (level-3 nodes, or leaves), e.g., PH.1.1, …, PH.1.50 are books belonging to PH.1 subcategory.

We simulated 2000 library checkout patterns, which come from four different classes of students. Each class of checkouts has its own distribution in checkout content; they are listed in table 1. Moreover, class I and II of checkouts are further divided into three and two subclasses, as shown in table 2 and 3, respectively. Totally, there are seven groups distributed differently in the library dataset.

Table 1：Simulative library dataset.

Class	# of checkouts	Content per checkout	Source
I	600	CS books: 80% MA books: 20%	Computer science students
II	500	MA books: 70% CS books: 20% PH books: 10%	Mathematics students
III	500	PH books: 80% MA books: 10% CH books: 10%	Physics students
IV	400	BI books: 80% CH books: 20%	Biology students

Table 2：The three subclasses of class I in simulative library dataset.

Subclass	Content per checkout		# of checkouts
	Under CS (80%)	Under MA (20%)	
I.1	CS.1: 90% CS.2: 10%	MA.1: 90% MA.2: 10%	250
I.2	CS.2: 90% CS.3: 10%	MA.2: 90% MA.3: 10%	200
I.3	CS.2: 10% CS.3: 10%, CS.4: 80%	MA.2: 10% MA.3: 10% MA.4: 80%	150

Table 3：The two subclasses of class II in simulative library dataset.

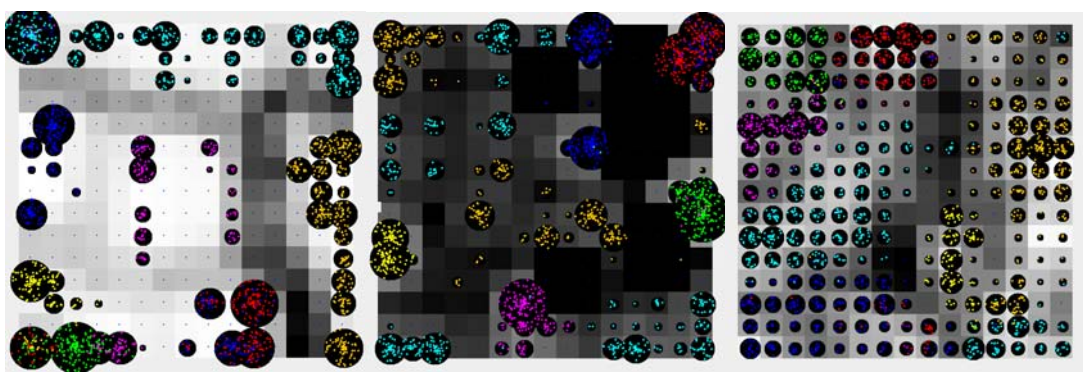
Subclass	Content per checkout			# of checkouts
	Under CS (20%)	Under MA (70%)	Under PH (10%)	
II.1	CS.1: 80% CS.2: 20%	MA.1: 80% MA.2: 20%	PH.1: 80% PH.2: 20%	300
II.2	CS.2: 20% CS.3: 80%	MA.2: 20% MA.3: 80%	PH.2: 20% PH.3: 80%	200

Each checkout of different classes is simulated by the following rules. Let T be the number of books in a checkout, and let X be a Poisson distribution with mean λ, i.e., $\lambda=E[X]$. Assume $T=X+1$, that is, T is determined by X. Besides, $E[T] = \lambda+1$. In our dataset, λ is set to nine. That is, in this simulative library, the average number of books

borrowed by students in each checkout is ten. Whenever the size of a checkout is determined, books in checkouts are decided by the checkout content distribution in different classes, which are listed in table 1, 2 and 3, respectively. For example, if a checkout belongs to class I.1, each book in this checkout has 72%, 8%, 18%, and 2% in probability from the subcategory CS.1, CS.2, MA.1, and MA.2, respectively.

4.2.1 Experimental result

Fig. 12 (a), (b) and (c) show the results of the library dataset mapped by the three comparing models, SetSOM, SCM, and SOM, respectively. Each map is displayed by a 15 by 15 U-Matrix (Ultsch 2003) as its background. In U-Matrix, the gray level in each neural unit represents the average distance between its own prototype and the prototypes of its nearest neighboring units. The larger the distance is, the darker the gray color is. It is reasonable the units with higher similarity in their prototype will gather in some lighter gray area. Therefore, the units of darker gray form a boundary of the lighter gray areas. We expect the input transactional patterns with higher similarity will be projected into some lighter gray area. After displaying the U-Matrix, we draw circles for units which have nonzero BMU count in foreground. The area of each circle depends on the unit's BMU count. We expect the input patterns with higher similarity will be projected into the same neuron or the neighboring neurons.



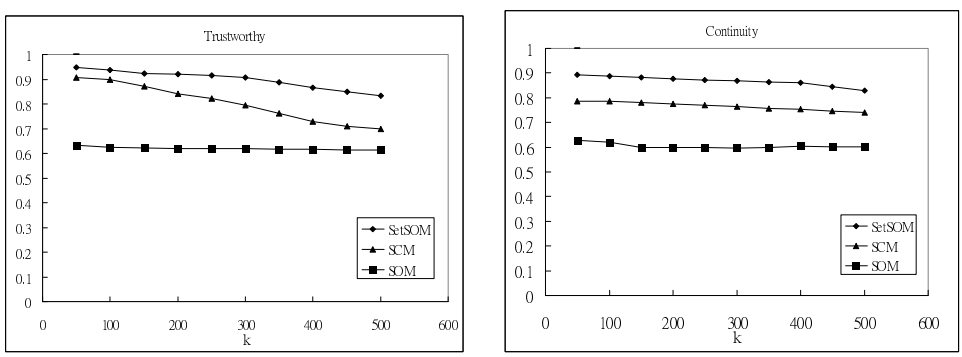
(a) (b) (c)

Figure 12：The mapping results of the three comparing models when using the simulative library dataset. (a) is the mapping results of the SetSOM, (b) is of the SCM, and (c) is of the SOM; each map has 15 by 15 units.

In order to know the mapping quality in visualization, we randomly scatter the

input patterns with specific colors within their BMU's circle. The colors of the input patterns are based on the classes to which they belong. Seven colors are assigned to the seven classes. In class I, the three subclasses of the checkouts are colored by red, green, and yellow, respectively. In class II, the two subclasses of the checkouts are colored by blue and magenta, respectively. The checkouts in class III and IV are colored by cyan and orange, respectively.

It is obvious the SetSOM has a better visualization and mapping quality than the others. In Fig. 12(a), the checkouts which belong to the same class are gathered as a group or distributed as nearer neighbors, and the structure of the U-Matrix perfectly coincides with the structure of our testing dataset. It is easy and precise to separate the SetSOM units into different groups with the aid of the U-Matrix. In fact, the gaps among units are clear without the U-Matrix. Besides, when comparing with the other models, the SetSOM gathers the three subclasses of class I and the two subclasses of class II closely than the other classes. The SetSOM reflects the actual structure of the dataset on the map more precisely than the others. In Fig. 12(b), the SCM can not exactly map the same class of the checkouts together and the boundaries of its U-Matrix are not very clear. In Fig. 12(c), the SOM has the same problems as the SCM, and the checkouts tend to be projected all over the neurons on the map.



(a) (b)

Figure 13：Mapping qualities of the three comparing models when using the simulative library dataset as an input. (a) for trustworthiness and (b) for continuity

Fig. 13 shows the trustworthiness and the continuity values versus the number of neighbors, k, varying from 50 to 500 when the simulative library dataset is mapped by the three comparing models. It is obvious the SetSOM outperforms than the others. All the values of the trustworthiness and the continuity of the SetSOM are greater than 0.80 and better than the others. That is, the neighborhood preservation and the local and global mapping quality of the SetSOM are better than the others.

4.3 Real dataset: articles of ACM proceedings

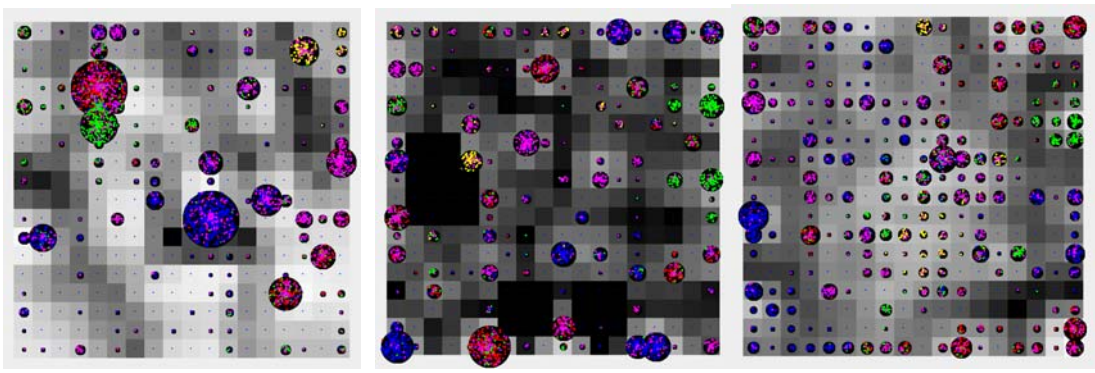
The real transactional dataset was extracted from the articles of five ACM proceedings including SIGIR, SIGMOD, SIGCHI, SIGMM, and SIGCOMM between 2000 and 2009. It is reasonable to consider each of the five ACM proceedings as a different class of articles. Based on the content, each article had been assigned some classification codes, which are considered as the transaction of the article. There are at least one primary classification code and a few additional classification codes. For example, "Comprehensive query-dependent fusion using regression-on-folksonomies: a case study of multimodal music search" (Zhang et al. 2009) is an article from the SIGMM, it contains three codes: {H.3.3.Query formulation, H.3.3.Search process, H.5.5.System}. For each article, the classification codes and its proceeding's name were extracted and saved. If an article contains only one code, we ignored it. At last, 3,357 transactions were collected. Table 4 gives the description of the real dataset in detail. All of the codes come from the ACM Computing Classification System (CCS); it is a four-level concept hierarchy. All the classification codes of the ACM articles and the CCS can be collected and downloaded from the ACM digital library.

Table 4：Articles of the five ACM proceedings

Proceeding	# of articles	Min. # of codes	Max. # of codes
SIGIR	689	2	10
SIGMOD	537	2	7
SIGCOMM	158	2	6
SIGCHI	920	2	18
SIGMM	1053	2	19

The mapping results of the articles by the three comparing models SetSOM, SCM, and SOM are depicted in Fig. 14(a), (b) and (c), respectively. Each map has 18 by 18

units and a U-Matrix displayed in the background. The boundaries are clear in the SetSOM. With the aid of the U-Matrix, we can separate the units into different groups in the SetSOM. Each article is mapped into its BMU with a specific color. Five different colors, red, green, yellow, blue, and magenta are assigned to SIGIR, SIGMOD, SIGCOMM, SIGMM, and SIGCHI, respectively. In the SetSOM, most of the articles belonging to the different proceedings can be gathered into different units. However, there exist many small units scattering over the map of the SetSOM. Moreover, quit a few articles of the SIGMM colored by magenta are mixed with the articles of the other proceedings in many units. This phenomenon emerges in both of the comparing models. In order to analyze this phenomenon, we removed the articles of SIGMM in the dataset and applied the SetSOM on the remained dataset. Fig. 15 depicts the result. Articles belonging to the different proceedings are almost projected into different units, although the units with the same color are not close together. It is reasonable to infer the topics in the SIGMM are more diverse than the other proceedings. Moreover, the topics in the SIGMM and the topics in the other proceedings are overlapping in some extent.



(a) (b) (c)

Figure 14： The mapping results of the ACM proceedings' articles. (a) is of the SetSOM, (b) is of the SCM, and (c) is of the SOM. Each map has 18 by 18 units.

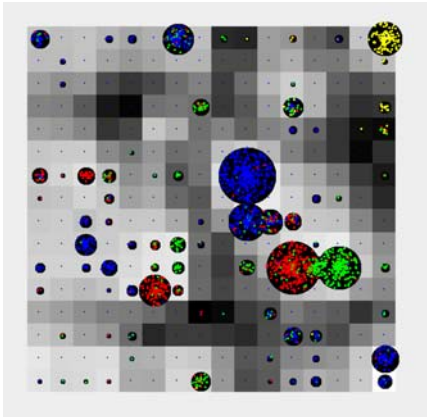


Figure 15 : The mapping results of the ACM proceedings' articles by the SetSOM, when the articles of the SIGMM are removed.

In order to validate the mapping result (Fig. 14(a)) of the SetSOM further, six of the neurons which locate at (4, 3), (10, 10), (4, 5), (12, 3), (11, 5), and (7, 8) in Fig. 14(a) were used to validate the projection result and to illustrate the differences of the articles among the neurons and the similarity of the articles inside the neurons. Note that (0, 0) is located at the upper left corner on the map. The first three neurons are the top three in BMU count; there are about 16.7%, 16.6%, and 7.0% of the articles mapped into the three neurons, respectively. The others are the top three in entropy value; there are about 2.7%, 0.2%, and 0.4% of the articles mapped into the three neurons, respectively.

For each of the six neurons, the classification codes of the articles inside the neuron were counted, and then the frequency distribution of the classification codes was drawn as a histogram. Fig. 16(a)-(f) depict the frequency distributions of the six neurons, respectively. All of the classification codes were only counted to the level-2 codes at most, for example, the three codes, H.2, H.2.1, and H.2.8.Data Mining, are all counted as the level-2 classification code, H.2. Furthermore, the classification codes were also counted according to what proceedings the articles belong to. Colors and styles in the histograms show the different proceedings.

In order to compare the distributions easily, the classification codes in the X-axes of Fig. 16(a), (b) and (c) are the same, and similarly in Fig. 16(d), (e), and (f). It is clear that the six neurons have very different kinds of distributions of their classification codes, that is, among the six neurons the types of their articles are quite different.

In Fig. 16(a) to (c), corresponding to the neurons (4, 3), (10, 10), and (4, 5), most of the classification codes concentrate on one level-2 classification code, that is, the

articles inside each of the three neurons must have high relevancy. We inspected the articles inside the neuron (4, 3), we found "information search, retrieval, indexing" is the main topic of the articles inside this neuron. Besides, 63.8% of the articles in neuron (4, 3) come from SIGIR; 78.6% of the articles in neuron (10, 10) come from SIGCHI; and 78.4% of the articles in neuron (4, 5) come from SIGMOD. However, the articles of SIGMM emerge in all these neurons. The neuron (4, 3) was inspected again; we found that those articles not coming from SIGIR are also relevant to information retrieval. For instance, the article "Comprehensive query-dependent fusion using regression-on-folksonomies: a case study of multimodal music search" (Zhang et al. 2009), which come from SIGMM, is obviously related to multimedia as well as information search and retrieval. Thus, it is reasonable this article is projected in this neuron.

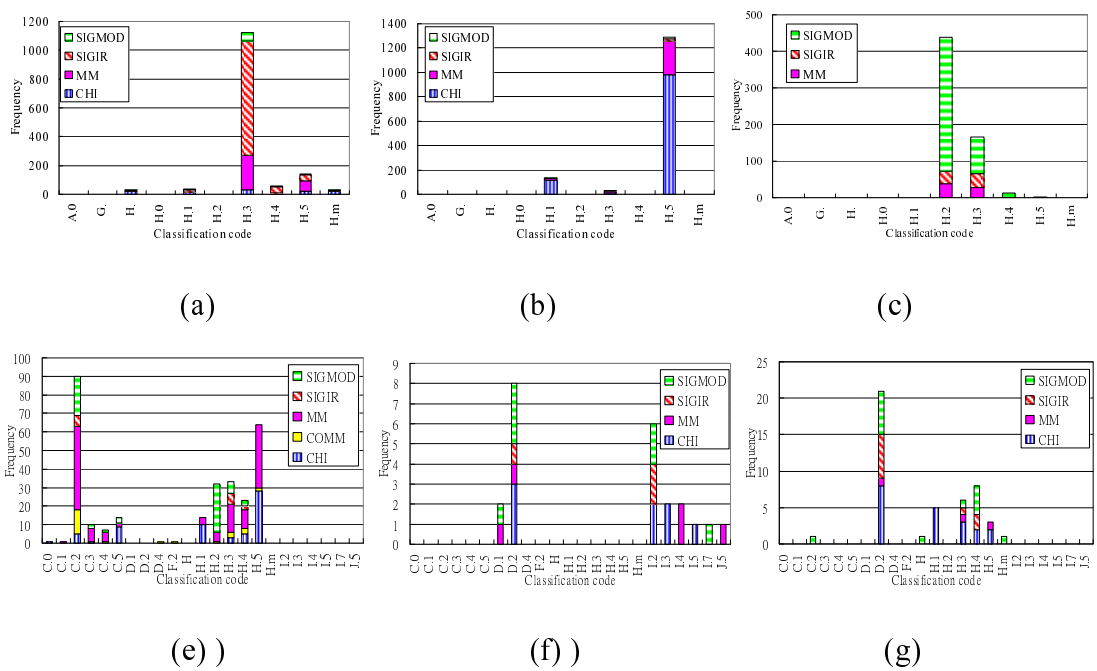
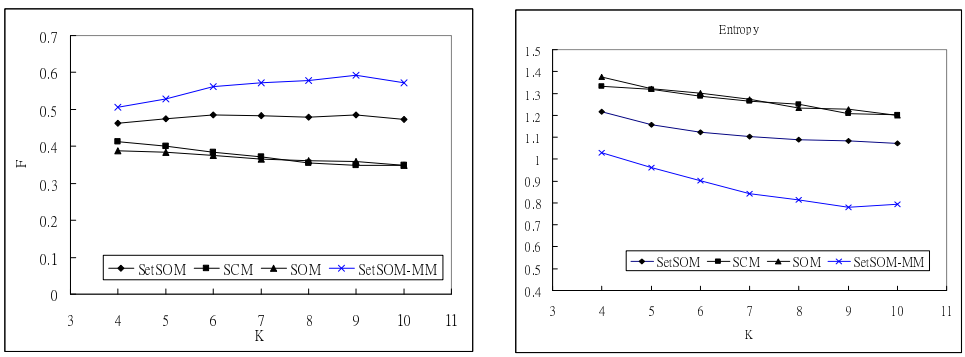


Figure 16：The frequency distributions of the classification codes of the selected neurons. (a), (b), (c), (d), (e), and (f) are the histograms of the neurons (4, 3), (10, 10), (4, 5), (12, 3), (11, 5), and (7, 8), respectively.

In Fig. 16(d) to (f), corresponding to the neurons (12, 3), (11, 5), and (7, 8), it is easy to find articles in these neurons come from diverse proceedings; this is why their entropy values are high. However, most of the articles have at least two classification

codes which are different in both of level-1 and level-2. For example, in the neuron (12, 3), C.2 and H.5 are the most common combination that the articles have. C.2 also combines with other H.1, H.2, H.3, or H.4. We found "Networked information systems' interfaces, retrieval, or database" is the main topic of the articles in this neuron. That is, high relevant articles are grouped in this neuron, though they may come from different proceedings.

Furthermore, we applied the K-Means to cluster the units with nonzero BMU count in the above four trained maps in Fig. 14 and 15 in order to inspect the clustering quality of the SetSOM preliminarily. The number of clusters was set from four to ten. We evaluated the clustering quality by the *F* measure and the weighted entropy. The *F* measures and the weighted entropy values of the four maps are depicted in Fig. 17(a) and 17(b), respectively. We found the SetSOM has a higher clustering quality than the other models, since its *F* measures are all higher than the other two models, and the weighted entropy values are all less than the others. Moreover, we found the clustering quality of the SetSOM map with SIGMM removed is much higher than the SetSOM with the original dataset. It coincides with the above inference about the SIGMM.



(a) (b)

Figure 17：(a) the F measures and (b) the entropy of clustering the SetSOM, SCM, SOM and the SetSOM without SIGMM by K-Means, K=4, …, 10.

5. Conclusions and future works

In this paper, we convert transactional data to tree structures and devise a distance function for the transactional data when an accompanied concept hierarchy exists. Based on the structure of concept hierarchies, the distance function takes the relevancy of items in transactions into account. Thanks to the tree-structured transactions and the

distance function, an extended SOM, the SetSOM, has been proposed to inspect the topological order of the transactional data. Via the adaptation method, imitating from trees growing in nature, the SetSOM has its good qualities in projecting transactional data into a 2-D map as well as in visualizing the inner structure of transactional data. From the experiments in the synthetic and the real datasets, the SetSOM outperforms other SOM models in visualization, mapping and clustering qualities. Our future work is to extend our model for transactional data no matter its concept hierarchy exists or not and to develop a clustering method for the trained SetSOM neurons in order to get a better clustering quality in transactional data.

6. Acknowledgement

This research was supported by the National Science Council, Taiwan, under grant NSC 100-2410-H-224-003-MY2.

Reference

Chen, D.R., Chang, R.F., and Huang, Y.L. (2000), 'Breast cancer diagnosis using self-organizing map for sonography', *Ultrasound in Medicine and Biology*, Vol. 1, No. 26, pp. 405-411.

Flanagan, J.A. (2003), 'Unsupervised clustering of symbol strings', Proceedings of the Int. Joint Conference on Neural Networks (IJCNN 2003).

Guha, S., Rastogi, R., and Shim, K. (2000), 'ROCK: a robust clustering algorithm for categorical attributes', *Information Science*, Vol. 25, No. 5, pp. 345–366.

Günter, S., and Bunke, H. (2002), 'Self-organizing map for clustering in the graph domain', *Pattern Recognition Letters*, Vol. 23, pp. 405–417.

Hammer, B., Micheli, A., Neubauer, N., Sperduti, A., and Strickert, M. (2005), 'Self-Organizing Maps for Time Series', *Proceedings of the Workshop on Self-Organizing Maps (WSOM 2005)*, Paris.

Han, J., and Kamber. M. (2001), *Data Mining Concepts and Techniques*, Morgan Kaufmann, San Francisco.

He, Z., Xu, X., and Deng, S. (2005), 'TCSOM: clustering transactions using self-organizing map', *Neural Processing Letters*, Vol. 22, pp. 249-262.

Himberg, J., Flanagan, J.A., and Mantyjarvi, J. (2003), 'Towards context awareness using Symbol Clustering Map', *Proceeding of WSOM (WSOM 2003)*.

Hsu, C.C. (2006), 'Generalizing self-organizing map for categorical data', *IEEE*

Transactions on Neural Networks, Vol. 17, pp. 294-304.

Hsu, C.C., Wang, K.M., and Wang, S.H. (2006), 'GViSOM for multivariate mixed data projection and structure visualization', *Proceedings of the Int. Joint Conference on Neural Networks (IJCNN 2006)*, pp. 3300-3305.

Hua Yan, H., Chen, K., Liu, L., and Bae, J. (2009), 'Determining the best K for clustering transactional datasets: A coverage density-based approach', *Data & Knowledge Engineering*, Vol. 68, pp. 28–48.

Kaski, S., Nikkila, J., Oja, M., Venna, J., Toronen, J., and Castren, E. (2003), 'Trustworthiness and metrics in visualizing similarity of gene expression', *BMC Bioinformatics,* Vol. 4, No. 48.

Kohonen, T. (1982), 'Self-organized formation of topologically correct feature maps', *Biological Cybernetics*, Vol. 43, pp. 59–63.

Kohonen, T. (1996), 'Self-organizing maps of symbol strings', Technical Report A42, Laboratory of Computer and Information Science, Helsinki University of Technology, Finland.

Kohonen, T. (2001), *Self-organizing maps*, Springer, 3rd extended ed., Berlin.

Kohonen, T., and Somervuo, P. (1998), 'Self-organizing maps on symbol strings', *Neurocomputing*, Vol. 21, pp. 19–30.

Kohonen, T., and Somervuo, P. (2002), 'How to make large self-organizing maps for nonvectorial data', *Neural Networks*, Vol. 15, pp. 945–952.

Kohonen, T., Hynninen, J., Kangas, J., and Laaksonen, J. (1996), 'SOM_PAK: the self-organizing map program package', Report A31, Helsinki University of Technology, Available from http://www.cis.hut.fi/research/som_pak/.

Kohonen, T., Kaski, S., Lagus, K., Salojarvi, J., Honkela, J., Paatero, V., and Saarela, A. (2000), 'Self-organization of a massive document collection', *IEEE Transactions on Neural Networks*, Vol. 11, No. 3, pp. 574-585,.

Kohonen, T., Oja, O.E., Simula, A. Visa, S.A., and Kangas, J. (1996), 'Engineering applications of the self-organizing map', *Proceedings of the IEEE (IEEE 1996)*,, Vol. 84, No. 10, pp. 1358-1384.

Lampinen, J., and Oja, E. (1992), 'Clustering properties of hierarchical self-organizing maps', *Journal of Mathematical Imaging and Vision*, Vol. 2, pp. 261-272.

Somervuo, P. (2004), 'Online algorithm for the self-organizing map of symbol strings', *Neural Networks*, Vol. 17, pp. 1231–1239.

Ultsch, A. (2003), 'Maps for the visualization of high-dimensional data spaces', *Proceedings of the 4th Workshop on Self-Organizing Maps (WSOM 2003)*,

Kitakyushu, Japan, Sep. pp. 225–230.

Vathy-Fogarassy, A., and Abonyi, J. (2009), 'Local and global mappings of topology representing networks', *Information Sciences*, Vol. 179, pp. 3791–3803.

Venna, J., and Kaski, S. (2005), 'Local multidimensional scaling with controlled tradeoff between trustworthiness and continuity', *Proceedings of the Workshop on Self-Organizing Maps (WSOM 2005)*, pp. 695–702.

Vesanto, J., Himberg, J., Alhoniemi, E., and Parhankangas, J. (2000), 'SOM Toolbox for Matlab 5', Report A57, Helsinki University of Technology, Available from http://www.cis.hut.fi/projects/somtoolbox/.

Wang, K., Xu, C., and Liu, B. (1999), 'Clustering transactions using large items', *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM 1999)*, pp. 483–490.

Yang, Y., Guan, S., and You, J. (2002), 'CLOPE: a fast and effective clustering algorithm for transactional data', *Proceedings of KDD (KDD 2002)*, pp. 682–687.

Yeh, M.F., and Chang, K.C. (2006), 'A self-organizing CMAC network with gray credit assignment', *IEEE Transactions on Systems, Man, and Cybernetics-Part B Cybernetics*, Vol. 36, No. 3, pp. 623-635.

Zhang, B., Xiang, Q., Lu, H., Shen, J., and Wang, Y. (2009), 'Comprehensive query-dependent fusion using regression-on-folksonomies: a case study of multimodal music search', *Proceedings of the 17th ACM international conference on Multimedia (MM 2009)*.