Applied Soft Computing 12 (2012) 3141–3157

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A self-organizing map for transactional data and the related categorical domain

Wen-Chung Liao, Chung-Chian Hsu*

Department of Information Management, National Yunlin University of Science and Technology, 123 University Road, Section 3, Douliou, Yunlin, Taiwan, ROC

ARTICLE INFO

Article history:
Received 5 February 2012
Received in revised form 12 June 2012
Accepted 19 June 2012
Available online 30 June 2012

Keywords:
Self-organizing map (SOM)
Transactional data
Categorical data
Mixed data
Data visualization
Distance measure
Tree-growing adaptation

ABSTRACT

After projecting high dimensional data into a two-dimension map via the SOM, users can easily view the inner structure of the data on the 2-D map. In the early stage of data mining, it is useful for any kind of data to inspect their inner structure. However, few studies apply the SOM to transactional data and the related categorical domain, which are usually accompanied with concept hierarchies. Concept hierarchies contain information about the data but are almost ignored in such researches. This may cause mistakes in mapping. In this paper, we propose an extended SOM model, the SOMCD, which can map the varied kinds of data in the categorical domain into a 2-D map and visualize the inner structure on the map. By using tree structures to represent the different kinds of data objects and the neurons' prototypes, a new devised distance measure which takes information embedded in concept hierarchies into consideration can properly find the similarity between the data objects and the neurons. Besides the distance measure, we base the SOMCD on a tree-growing adaptation method and integrate the U-Matrix for visualization. Users can hierarchically separate the trained neurons on the SOMCD's map into different groups and cluster the data objects eventually. From the experiments in synthetic and real datasets, the SOMCD performs better than other SOM variants and clustering algorithms in visualization, mapping and clustering.

1. Introduction

The self-organizing map (SOM) is excellent in mapping and visualizing high dimensional data since it was proposed by Kohonen [1,2]. The SOM projects high dimensional data objects into a lattice of neurons, typically arranged on a two-dimension map; in the meanwhile, the topological order among the data objects in the data space can be preserved on the 2-D map. Users of the SOM can easily view the inner structure of high dimensional data on a 2-D map.

Based on the use of batch-SOM training and the median associated with each neuron, the SOM can be applied to handle non-vector data with a predefined distance measure or similarity [3–5]. Later, the online SOM algorithms for non-vector data such as symbol string [6–8], graph [9], transactions [10], time series [11], etc. had been developed.

No matter what kinds of data, once they are projected into a 2-D map by the SOM, users can easily investigate and visualize their inner structures on the map. In the early stage of data mining, it is important for any kind of data to inspect their inner structure. However, very few studies apply the SOM to transactional data and the related categorical domain.

The categorical domain we mean include any kind of data which contain categorical or nominal values. Transactional data, categorical data, and mixed data are the three usual kinds of data which belong to the categorical domain. Each data instance in transactional data is a set of items and each item is a categorical or nominal value. In the case of categorical data, each data instance has a fixed number of categorical attributes. Each categorical attribute has some possible categorical values, but only one of the possible values will be the attribute value in any data instance. If the attributes of each data instance mix with categorical and numerical ones, it is the so-called mixed data.

Since categorical values have no geometrical meanings, it is not easy to inspect their data structures or how they distribute via the data mining tools which are usually used in numerical data well. So far, few SOM studies had been proposed to mine the categorical domain. The TCSOM [10] can cluster transactional data but provides no map for users to view the inner structure of transactional data, the GSOM [12] can inspect the inner structure of mixed data but cannot view their hierarchical structures more precisely, the SCM [6,7] can handle transactional data but ignores "relevancy information" between items, which we explain later. The above SOM models have some disadvantages in mining the categorical domain.

When we mine the categorical domain, transactional data are especially difficult to handle since transactional data are non-vector set type data. Unlike categorical and mixed data, transactional data are unstructured data. One of the most typical transactional data

* Corresponding author. Tel.: +886 5 5342601x5326; fax: +886 5 5312077.
E-mail address: hsucc@yuntech.edu.tw (C.-C. Hsu).

1568-4946/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.asoc.2012.06.014

in our daily life is the market basket data, which has its importance in any marketplace. The market basket data of any shopping store always reveal the consumers' shopping behaviors and personal favorites. Clustering the market basket data can separate the consumers into different segments, so the managers of the shopping store can adopt different marketing strategies for the different segments of the consumers to maximize their profits.

In many application domains, transactional data are accompanied with a concept hierarchy or a taxonomy tree. The product catalog of a shopping store is a typical example of a concept hierarchy. Concept hierarchies are tree structures [13]. All of the possible items in a transactional database form the leaf nodes of the accompanying concept hierarchy. The non-leaf nodes are the categories (or concepts) of all the possible items. The relevancy information among all of the possible items in the transactional data is embedded in the concept hierarchy. Items of the same category are more relevant than those of items in different ones. For example, in a shopping mall, the apple juices are more relevant to the grape juices than the apples, since the apple juices and the grapes juices belong to the same juice category and the apples are in the fruit category. If the apple juices are in shortage on the shelf, customers normally tend to take a grape juice rather than an apple.

So far, as we know, there are no SOM models that consider concept hierarchies when handling transactional data. Neither the SCM [6,7] nor the TCSOM [10] considers the existing of concepts hierarchies. The GSOM [12] and the GViSOM [14] were proposed to handle mixed-type data. Although they take concept hierarchies, which are associated with each categorical attribute, into consideration, they can not apply to transactional data well since their data objects are fixed dimensional vectors but not sets.

Even most of transactional data clustering algorithms [15–18] ignore the existing of concept hierarchies. The relevancy between individual items is never considered in most of the algorithms. Most of them put the transactions which have more items in common into the same cluster. 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 Large Items [16], CLOPE [17], SCALE [18,19], would not put these two transactions into the same cluster, since these two transactions do not have any items in common. It is clear that these two transactions are similar since their items are highly relevant.

In this paper, an extended SOM model, named as SOMCD, is proposed for the categorical domain, which is usually accompanied with concept hierarchies. Different kinds of data, including transactional data, categorical data, and mixed data, all related to the categorical domain can be projected into a lattice of neurons arranged on a two-dimensional map by the SOMCD; in the meanwhile, the topological order of the data can be preserved and visualized on the 2-D map. By using tree structures to represent the varied kinds of data objects and the prototypes of the neurons, a new devised distance measure can take the relevancy information embedded in the concept hierarchies into consideration and find the similarity between the data objects and the neurons properly. Besides the new devised distance measure, we base the SOMCD on a tree-growing adaptation method and integrate the U-Matrix for visualization.

This study has the following contributions. The first one is the SOMCD extends the application scope of the conventional SOM to include the transactional data which are accompanied with a concept hierarchy. The second is a distance function is devised for the categorical domain such that the relevancy information embedded in concept hierarchies can be measured. The third is the SOMCD is a total solution in projecting, visualization, and clustering for any kind of data which are related to the categorical domain.

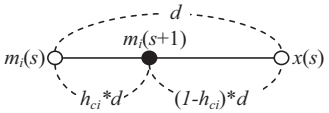


Fig. 1. Adaptation of reference vector.

The rest of this paper is organized as follows. In Section 2, the SOM and some extended models are reviewed. Section 3 gives an abstract data structure to represent the varying types of data all related to the categorical domain and a distance function to measure the dissimilarity between them. Algorithms of the SOMCD are described in Section 4. Experiments are reported in Section 5. Conclusions and future works are given in Section 6.

2. Related works

2.1. Self-organizing map

The self-organizing map is an unsupervised neural network [1,2]. 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 the 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 the 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 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 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:

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

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

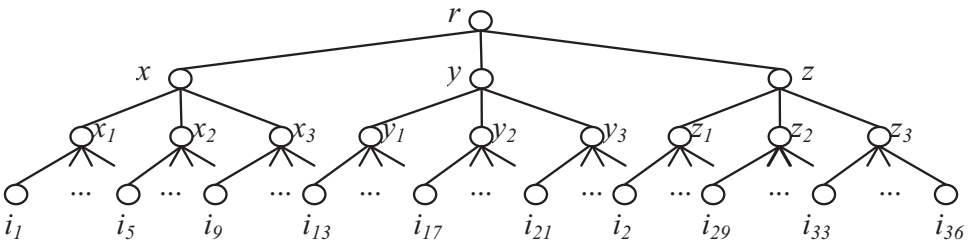


Fig. 2. A three-level concept hierarchy. The nodes x, y, and z are the main categories; the nodes x_is, y_is, and z_is are the subcategories; the i_js are the items.

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

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

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

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

2.2. SOMs on the categorical domain

Some of the SOM models which can handle the categorical domain are reviewed in this subsection. The Symbol String Clustering Map (SCM) is an unsupervised clustering method for symbol strings [6,7]. 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 in the neighborhood of the winner node, both of its node string and weight vector are adjusted according to the input symbol string, the winner node, the values of the learning rate and the kernel function at time t. After several times of training, the SCM groups the non-null nodes into many clusters and separates the clusters by the null nodes on the map. Transactions are symbol strings if their items are considered as symbols. Thus, we can apply the SCM on transactional data. But the SCM does not consider any information embedded in the accompanying concept hierarchies of transactional data. The mixed data is beyond the scope of the SCM.

The TCSOM [10] is a SOM model which can cluster transactional data. In the TCSOM, the number of the neurons M must be selected at the first beginning and the M neurons are the M clusters of transactional data at the end. The TCSOM treats each transactional data as a binary vector, and assigns each neuron a weight vector. A normalized dot product norm is devised for measuring the distance between input vectors and neurons' weight vectors. Whenever a winner neuron is found for each data instance, only the winner's weight vector is updated. After scanning the dataset one pass, the TCSOM maps the transactional data into the M neurons. Therefore, the M clusters are obtained. The TCSOM is efficient but does not have a 2-D map, so the visualization ability is lost. The TCSOM also ignores any information embedded in the accompanying concept hierarchy of transactional data and is not suitable for mixed data as well.

The generalizing self-organizing map (GSOM) [12] can properly reflect the correct topological order of mixed data on a 2-D map. The GSOM associates each categorical attribute with a distance hierarchy, a special kind of concept hierarchy. The relevancy information between the categorical values of each categorical attribute is captured by the distance hierarchies, so the GSOM can distinguish the similarity between data instances with mixed type values. However, the GSOM cannot deal with transactional data, since the data objects are treated as fixed dimensional vectors not sets of items. Later, the GSOM integrates the ideas from the ViSOM [20] to overcome the shortage in projection distortion on the map [14]. Like the GSOM, this new extended SOM, named as GViSOM [14], focus on mixed data only.

3. Distance function on categorical domain

In this section, we introduce a distance function which can be used in varied types of data all related to the categorical domain. Even though missing values occur in data, the distance function works properly. First of all, we devise a distance function on concept trees which are the data structures we propose for the varied types of data. Based on the new devised distance function, the distance functions on the varied types of data will be given.

3.1. Distance function on concept trees

Let $I = \{i_1, i_2, \ldots, i_M\}$ be a set of M possible items (or categorical values). Let H be a concept hierarchy of I. It means that H must be a rooted and labeled tree with all the 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 non-leaf nodes in H as the *category nodes* and the leaves as the *item 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. For example, a three-level concept hierarchy is depicted in Fig. 2. If a concept hierarchy has item nodes without category nodes, we call it as a *type-0* concept hierarchy; otherwise, we call it a *type-1* concept hierarchy. It is obvious a type-1 concept hierarchy contains more relevancy information between the items than a type-0 one.

Let T be a subtree of H and let a denote a node in T. A value denoted by $g_T(a)$ is given to node a and is called the *growth value* of 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. The weight of T, denoted by $|T|$, is the total growth values of all the nodes in T, that is, $|T| = \Sigma_{a \in T} g_T(a)$. The edges in T are only used to connect the nodes in T. In this study, whenever two adjacent nodes in H are in T, the edge between the two nodes is in T. 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 in H, these nodes must be leaves in T. If the root node of T is fully grown, then T is called a *concept tree* of H. A concept tree represent a concept, i.e., the root node of T, has been developed so far. All the nodes in a concept tree T are the current content of the concept. So, we define the net weight of a concept tree T, denoted by $||T||$, as the weight of the content of T, that is, the weight of T minus the growth value of the root node. Thus, we have $||T|| = |T| - 1$. If r is in T, T is called an *r-concept tree*. In any concept tree, we confine only the root node and the fully grown category nodes can have child nodes.

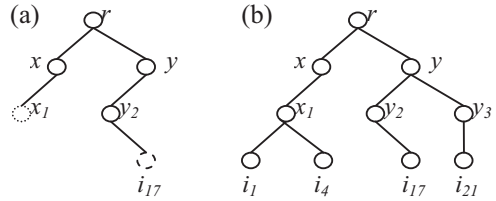


Fig. 3. Examples of an *r*-concept tree and a transaction tree. (a) An *r*-concept tree; (b) the transaction tree *T* of $\{i_1, i_4, i_{17}, i_{21}\}$ with $||T|| = 9$.

Let *t* be a set of items from *I*. That is, *t* is a subset of *I*. In the terminologies of transactional data, *t* is called a *transaction* with its items from *I*. If *T* is an *r*-concept tree, and all the leaves of *T* are fully grown and exactly equal to the items in *t*, then *T* is called the corresponding *transaction tree* of *t*. It can be proved easily that *T* is unique.

Based on the concept hierarchy in Fig. 2, an example of an *r*-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 we have $t = \{i_1, i_4, i_{17}, i_{21}\}$, then the transaction tree *T* with $||T|| = 9$, which corresponds to *t*, is depicted in Fig. 3(b).

An *r*-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. *r*-Concept trees will be used as the prototypes of the neurons in our SOM model.

Before we have a distance function defined on concept trees, here are some notations. For any concept tree *T*, $\pi_T(b)$ is denoted as the set of all the child nodes of *b*, and $\Psi_T(a|b)$ is denoted as the largest subtree of *T* rooted at the node *a* with *b* as *a*'s parent node, if no ambiguity, *b* can be ignored. Note that if *a* is fully grown, $\Psi_T(a|b)$ is a concept tree. Based on the Jaccard's coefficient[1] [13], a distance function of any two concept trees *T* and *U* with a fully grown node *b* as their common root is given as follows:

$$\mathrm{dist}(T, U) = \frac{W\mathrm{diff}(T, U)}{W\mathrm{intersect}(T, U) + W\mathrm{diff}(T, U)}, \tag{7}$$

where

$$W\mathrm{diff}(T, U) = \sum_{a \in \pi_T(b) - \pi_U(b)} |\Psi_T(a|b)| + \sum_{a \in \pi_U(b) - \pi_T(b)} |\Psi_U(a|b)| + \sum_{a \in \pi_T(b) \cap \pi_U(b)} d(\Psi_T(a|b), \Psi_U(a|b)), \tag{8}$$

$$W\,\mathrm{intersect}(T, U) = \sum_{a \in \pi_T(b) \cap \pi_U(b)} \frac{|\Psi_T(a|b)| + |\Psi_U(a|b)| - d(\Psi_T(a|b), \Psi_U(a|b))}{2}, \tag{9}$$

and

$$d(\Psi_T(a|b), \Psi_U(a|b)) = \begin{cases} \left| |\Psi_T(a|b)| - |\Psi_U(a|b)| \right| & \text{if } \Psi_T(a|b) \text{ or } \Psi_U(a|b) \text{ degenerates to node } a, \\ \mathrm{dist}(\Psi_T(a|b), \Psi_U(a|b)) & \text{otherwise.} \end{cases} \tag{10}$$

Note that $W\mathrm{diff}(T, U)$ and $W\mathrm{intersect}(T, U)$ 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 $d(\Psi_T(a|b), \Psi_U(a|b))$ is used to calculate the difference between the two subtrees $\Psi_T(a|b)$ and $\Psi_U(a|b)$. If either $\Psi_T(a|b)$ or $\Psi_U(a|b)$ degenerates to a node *a*, we directly calculate their weight difference as their difference. Otherwise, we recursively use the $\mathrm{dist}(\cdot,\cdot)$ to calculate the distance between $\Psi_T(a|b)$ and $\Psi_U(a|b)$ as their difference. In the latter case, we tend to treat $\Psi_T(a|b)$ and $\Psi_U(a|b)$ 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 $\Psi_T(a|b)$ and $\Psi_U(a|b)$ are much bigger than the distance.

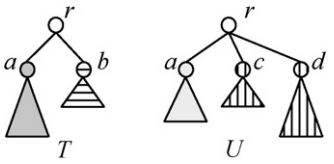


Fig. 4. Two *r*-concept trees and their first level subtrees.

We further explain the distance function by an example in Fig. 4. Suppose *T* and *U* are two *r*-concept trees. From the point of view of the root nodes, there are three different parts between *T* and *U*. (i) $|\Psi_T(b|r)|$: since $\Psi_T(b|r)$ is completely in *T* and not in *U*. (ii) $|\Psi_U(c|r)| + |\Psi_U(d|r)|$: since the subtrees $\Psi_U(c|r)$ and $\Psi_U(d|r)$ are completely in *U* and not in *T*. (iii) $\mathrm{dist}(\Psi_T(a|r), \Psi_U(a|r))$: although $\Psi_T(a|r)$ and $\Psi_U(a|r)$ are two similar subtrees, differences may occur inside the two subtrees $\Psi_T(a|r)$ and $\Psi_U(a|r)$. Thus we recursively use the distance function to measure their differences. Therefore, by the definition of $W\mathrm{diff}(\cdot,\cdot)$, the mutual differences between *T* and *U* are the total of the three parts as follows:

$$W\mathrm{diff}(T, U) = |\Psi_T(b|r)| + |\Psi_U(c|r)| + |\Psi_U(d|r)| + \mathrm{dist}(\Psi_T(a|r), \Psi_U(a|r)). \tag{11}$$

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 $W\mathrm{intersect}(\cdot,\cdot)$, the intersection between *T* and *U* is calculated by the following average:

$$W\mathrm{intersect}(T, U) = \frac{|\Psi_T(a|r)| + |\Psi_U(a|r)| - \mathrm{dist}(\Psi_T(a|r), \Psi_U(a|r))}{2}. \tag{12}$$

When such a distance function applied on concept trees, it can measure the differences in their tree structures, since the distance function recursively compare the structures from their root nodes to the leaf nodes.

3.2. The distance function on transactional data

Let $D = \{t_1, \ldots, t_N\}$ be a transactional dataset. Each t_i is called a transaction of *D* and is a set of distinct items. Assume $I = \{i_1, i_2, \ldots, i_M\}$ is the set of all the possible items in the transactions of *D*. Then each t_i is a subset of *I*. If there exists a concept hierarchy *H* of *I*, then *H* is called an accompanying concept hierarchy of *D*. If *H* is a type-1 concept hierarchy, we call *D* a type-1 transactional dataset. Of course, for each t_i in *D*, there is a corresponding transaction tree T_i from *H*.

[1] Jaccard's coefficient of any two finite sets A, B is defined as $|A \cap B|/|A \cup B|$.

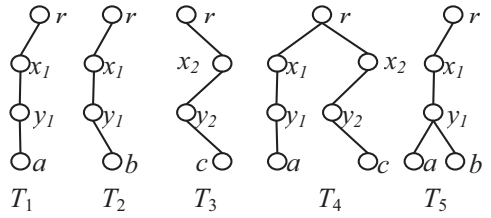


Fig. 5. Transaction trees of t_1, t_2, t_3, t_4, and t_5.

Now, based on the above distance function defined on the concept trees, we can easily give a distance measure between any two transactions t_i and t_j in D as follows:

$$\text{dist}(t_i, t_j) = \text{dist}(T_i, T_j), \tag{13}$$

where T_i and T_j are the corresponding transaction trees of t_i and t_j, respectively.

The distance 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 five transactions $t_1 = \{a\}$, $t_2 = \{b\}$, $t_3 = \{c\}$, $t_4 = \{a, c\}$, and $t_5 = \{a, b\}$ whose items are leaves of some concept hierarchy, and their corresponding transaction trees are depicted in Fig. 5, respectively. Then, we have $\text{dist}(t_1, t_2) = 1/8$, $\text{dist}(t_1, t_3) = 1$, $\text{dist}(t_2, t_3) = 1$, $\text{dist}(t_1, t_4) = 1/2$, and $\text{dist}(t_1, t_5) = 4/79$ by applied the above distance function. If the traditional Jaccard's coefficient is used, then the relationship of the items in the concept hierarchy is ignored, and we have $\text{dist}(t_1, t_2) = 1$, $\text{dist}(t_1, t_3) = 1$, $\text{dist}(t_2, t_3) = 1$, $\text{dist}(t_1, t_4) = 1/2$, and $\text{dist}(t_1, t_5) = 1/2$. The similarity among t_1, t_2, t_3, t_4, and t_5 cannot be identified, since t_1 is similar to t_2 than t_3 and much similar to t_5 than t_2.

If the accompanying concept hierarchy H of D is a type-0 concept hierarchy, that is H has no category nodes and is a one level concept hierarchy with all the possible items in D as its leaves, then we call D a type-0 transactional dataset. This is a special case of the type-1 transactional data. Let t and u be any two transactions in D, and let T and U denote the corresponding transaction trees of t and u. T and U are one level concept trees. If Eqs. (7)–(10) are applied to such transactions, the distance between them can be simplified as follows:

$$\begin{aligned}\text{dist}(t, u) &= \frac{\left|\pi_T(r) - \pi_U(r)\right| + \left|\pi_U(r) - \pi_T(r)\right|}{\left|\pi_T(r) \cup \pi_U(r)\right|} \\ &= \frac{|t - u| + |u - t|}{|t \cup u|} = 1 - \frac{|t \cap u|}{|t \cup u|}\end{aligned} \tag{14}$$

It is shown that the distance function we give coincides with the Jacarrd's coefficient in this special case.

In the distance calculation of the type-0 transactional data, the accompanying concept hierarchy is not important and useless. Since the concept hierarchy does not provide any relevant information among items in the type-0 transactional data.

3.3. The distance function on categorical data

In this subsection, the distance function defined in Eqs. (7)–(10) will be applied to categorical data. Let D be a dataset with p categorical attributes $A_1, A_2, \ldots, A_p$. Suppose each attribute A_i has r_i possible categorical values and the r_i possible values are the leaves of a concept hierarchy CH_i. Then we can create an associated concept hierarchy H, depicted in Fig. 6(a), for the categorical data by the following rules:

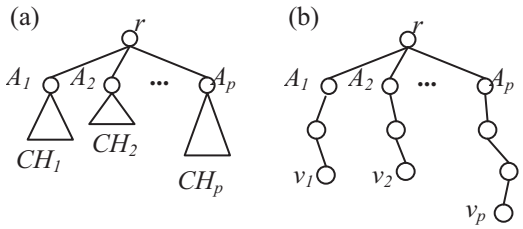


Fig. 6. (a) A concept hierarchy for categorical data and (b) the transaction tree of a categorical data object $(v_1, v_2, \ldots, v_p)$.

- Let $A_1, A_2, \ldots, A_p$ be the level-1 category nodes of H.
- For each category node A_i, grow a concept hierarchy CH_i with A_i as its root.

Let $v = (v_1, v_2, \ldots, v_p)$ be a data object in D. Based on the concept hierarchy H, v can be easily transformed to a transaction tree V as follows, if v is regarded as a transaction. For each attribute node A_i in V, there grows a concept tree V_i rooted at A_i. V_i must have v_i as its only one fully grown leave and is just like the tree depicted in Fig. 6(b). Therefore, the distance function defined in Eqs. (7)–(10) can be applied to categorical data. Let u be another data object in D with U as its corresponding transaction tree. Then for each attribute node A_i in U, there must exist a concept tree U_i rooted at A_i with u_i as its only one leave. Assume $\text{dist}(U_i, V_i) = d_i$, then the distance between u and v can be simplified as follows:

$$\text{dist}(u, v) = \frac{\sum d_i}{\sum (|U_i| + |V_i| + d_i)/2} \tag{15}$$

If no missing values, the maximal distance value is $2p/(|U| + |V| + p)$, whenever $d_i = 1$ for all i, and the minimal distance value is 0, whenever $d_i = 0$ for all i. When divided by $2p/(|U| + |V| + p)$, the distance values can be normalized. If missing values exist, the distance function can be applied properly.

If all the concept hierarchies of the categorical attributes are type-0, we have a type-0 categorical dataset. Otherwise, we have a type-1 categorical dataset. Type-0 categorical dataset is a special case of type-1. For any data object v in the type-0 categorical dataset, v can be transformed to a two-level concept tree with the p categorical attributes reserved in the level-1 nodes. In many transactional data clustering algorithms [10,15,17], v is transformed to a transaction, a set of the categorical values in v, or a binary vector, the attribute schema is lost.

3.4. The distance function on mixed data

In this subsection, the distance function defined in Eqs. (7)–(10) will be applied to mixed data. Let D be a mixed dataset with p categorical attributes $A_1, A_2, \ldots, A_p$ and q numerical attributes $B_1, B_2, \ldots, B_q$. Suppose each categorical attributes A_i has r_i possible categorical values and is accompanied with a concept hierarchy CH_i. Then we can create a concept hierarchy H for the mixed data as follows:

- Let $A_1, A_2, \ldots, A_p, B_1, B_2, \ldots, B_q$ be the level-1 category nodes of H.
- For each category node A_i, grow a concept hierarchy CH_i with A_i as its root node.
- Each B_i, grow a virtual node b_i as its only one leaf node.

Let $v = (v_1, v_2, \ldots, v_p, v_{p+1}, v_{p+2}, \ldots, v_{p+q})$ be a data object of D. Based on the concept hierarchy H, v can be easily transformed to an r-concept tree V as follows. For each attribute node A_i in V, there grows a concept tree V_i rooted at A_i with v_i as its only one fully

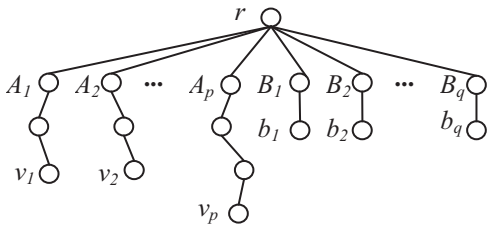


Fig. 7. A transaction tree of a mixed-type data object.

grown leave. For each attribute node B_j in V, $j = 1, \ldots, q$, a child node b_j will be grown with the following growth value:

$$g_V(b_j) = F_{B_j}(v_{p+j}), \tag{16}$$

where $F_{B_j}(.)$ is the cumulative distribution function (CDF) of the values in attribute B_j. The simplest CDF is the uniform distribution function, that is,

$$g_V(b_j) = \frac{v_{p+j} - \min(B_j)}{\max(B_j) - \min(B_j)}, \tag{17}$$

where $\max(B_j)$, $\min(B_j)$ are the maximum and minimum of the values in attribute B_i. The normal distribution function is another CDF we can use; it depends on the distribution of the values in attribute B_j. Thus, the graph of V is depicted in Fig. 7. If no ambiguity, V is still called as a transaction tree for the data object v.

Therefore, the distance function defined in Eqs. (7)–(10) can be applied to mixed data. Let u be another data object in D with U as its corresponding transaction tree. Then for each attribute node in U, there grow a concept tree U_i. Assume $\text{dist}(U_i, V_i) = d_i$, for $i = 1, \ldots, p$, and $\text{dist}(U_{p+j}, V_{p+j}) = e_j$, for $j = 1, \ldots, q$, then the distance between u and v can be simplified as follows:

$$\text{dist}(u, v) = \frac{\sum d_i + \sum e_j}{\sum(|U_i| + |V_i| + d_i)/2 + \sum(g_U(b_j) + g_V(b_j) + e_j)/2} \tag{18}$$

4. SOMCD

In this section, a SOM model for the categorical domain is proposed. We call it SOMCD. Before we propose the SOMCD, three operators on concept trees are introduced.

4.1. Operators on concept trees

In order to modify concept trees in our SOM model, we extend three set operators, union, intersection and difference, on concept trees with the same root node. Let T and U be two concept trees of H with the same root node. $T \cap U$ 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 the 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 a concept tree with the nodes in T or U. Let a be a node in $T \cup U$, the growth value of a is determined as follows:

$$g_{T\cup U}(a) = \begin{cases} g_T(a) & \text{if } a \in T \text{ and } a \notin U \\ g_U(a) & \text{if } a \notin T \text{ and } a \in U \\ \max(g_T(a), g_U(a)) & \text{if } a \in T \text{ and } a \in U \end{cases} \tag{19}$$

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

$$g_{T-U}(u) = \begin{cases} g_T(a) & \text{if } a \in T \text{ and } a \notin U \\ g_T(a) - g_U(a) & \text{if } a \in T,\ a \in U,\ \text{and } g_T(a) > g_U(a) \end{cases} \tag{20}$$

The result of $T - U$ could be a forest of subtrees in T and not in U. For example, if U is a concept tree whose root node r has one partially grown child node w_1, and T is a concept tree whose root node r has three fully grown child nodes 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. They may not be concept trees.

Let T and U be any two concept trees with a common root node from H, then we have the following properties: (i) $||T \cup U|| = |U - T| + |T - U| + ||T \cap U||$, (ii) $||T \cup U|| = ||T|| + |U - T|$, and (iii) If both T and U are one-level concept trees with a common root node, then, we have

$$\text{dist}(T, U) = \frac{\left|T - U\right| + \left|U - T\right|}{\left\|T \cup U\right\|} = 1 - \frac{\left\|T \cap U\right\|}{\left\|T \cup U\right\|}. \tag{21}$$

Follow the definitions of the above three operators, the two properties (i) and (ii) can be proved. Follow Eqs. (7)–(10) and the definitions of the three operators, the property (iii) can be proved. Eq. (21) coincides with the Jaccard's coefficient.

4.2. SOMCD and its algorithms

In this subsection, we introduce the SOMCD, which can handle the varied types of data all related to the categorical domain. That is, the dataset D we use in the SOMCD could be any type of data in Section 3. No matter which type of data we use, we have an accompanying concept hierarchy H of D, and each data object in D can be transformed to a transaction tree.

4.2.1. Initialization phase

The SOMCD 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 SOMCD is an r-concept tree from 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 SOMCD will be initially assigned an r-concept tree as its prototype. Those r-concept trees can be randomly generated based on the concept hierarchy H. For convenient sake, for each neuron, a data object is randomly drawn from the dataset D, and then its corresponding transaction tree is simply assigned as the neuron's prototype.

4.2.2. Training phase

In this phase, the SOMCD learns from input patterns randomly and iteratively. The input patterns are data objects from the dataset D. Once a data object randomly drawn from D, the SOMCD 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 neurons [21,22].

The same as the traditional SOM models, there are two key steps whenever a data object 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 neurons which are in a neighborhood of the BMU. In the first step, if t is the data object randomly chosen from the dataset D and the corresponding

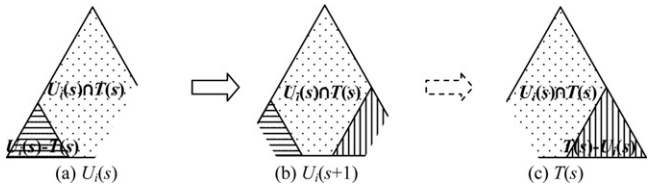


Fig. 8. Adaptation of the concept tree $U_i(s)$ of neuron i when $T(s)$ is given at time s.

transaction tree T of t is fed to the SOMCD, the BMU c is found as follows:

$$c = \arg \min_i \ \text{dist}(T, U_i), \tag{22}$$

where U_i is the concept tree of the ith neuron in the SOMCD. That is, the BMU is the neuron which has the smallest distance between its concept tree and the given transaction tree.

Then followed by the second key step, neurons in the neighborhood of the BMU learn from the data object 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 neuron i at the time s. According to the adaptation mechanism of the conventional SOM, which we have described in Section 2, $U_i(s)$ should be adjusted toward $T(s)$ such that the distance between them is decreased. That is, if $\text{dist}(T(s),U_i(s)) = d$, $U_i(s+1)$, the concept tree of neuron i after adaptation, must satisfy the following two equations:

$$\text{dist}(T(s), U_i(s+1)) = d \cdot (1 - h_{ci}(s)), \tag{23}$$

$$\text{dist}(U_i(s), U_i(s+1)) = d \cdot h_{ci}(s), \tag{24}$$

where $h_{ci}(s)$ is the value of the neighborhood function at the time s. That is, $d \times h_{ci}(s)$ 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 exactly satisfy both of Eqs. (23) and (24). The similar case happens in the study of Günter and Bunke [9].

Now, we describe the way how we adjust the concept tree $U_i(s)$ of neuron i such that $U_i(s+1)$ can get more similar to $T(s)$ at time s. The adaptation method we adopt imitates trees growing in nature. For convenient sake, the concept trees $U_i(s)$ and $T(s)$ are simplified to Fig. 8(a) and (c), respectively. The areas filled with little dots in Fig. 8(a) and (c) are the common parts of $U_i(s)$ and $T(s)$. The area filled with horizontal lines in Fig. 8(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. 8(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 $U_i(s)$. The input pattern $T(s)$ can be considered as the total conditions (sunlight, water, temperature, nutrition and so on) that environment currently provides to trees. Trees follow the environmental pattern to grow.

Therefore, a reasonable way to adjust the concept tree $U_i(s)$ of 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 $U_i(s)$. Let $W_{pos} = |T(s) - U_i(s)|$ and $W_{neg} = |U_i(s) - T(s)|$. 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. 8(b) represents the concept tree of neuron i after adaptation. Once the percentages h_1 and h_2 are determined, $W_{pos} \times h_1$ 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 the negative adjustment values, 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. 9, 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 $U_i(s)$, 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.

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 Eqs. (23) and (24). 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 $U_i(s)$, in the meanwhile $U_i(s+1)$ gets closer to $T(s)$.

4.2.3. Adaptation algorithms of SOMCD

Now the entire algorithm of the adaptation step is shown in Fig. 10. 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 the BMU is determined on the map. For each neuron i in the neighborhood of the BMU, $U_i(s)$ is modified according to the input pattern $T(s)$ as well as the value of the neighborhood function $h_{ci}(s)$. During each modification process, the positive part, the negative part, and the growing points are determined for neuron i. It is easy to find out the subtrees in both of the positive and the negative part 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.

Then by using the ManySubtreesGrow algorithm, depicted in Fig. 11, 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 ManySubtreesGrow algorithm, so we omit it. Similarly, by using the ManySubtreesPrune algorithm, which is depicted in Fig. 12, 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.

4.3. Visualization of the SOMCD

The visualization of the SOMCD is based on the U-Matrix [23,24]. The 2-D map of the SOMCD is displayed by an m by n neural units with a U-Matrix as its background. In a 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. The units of darker gray form a boundary of the lighter gray areas. It is reasonable the units with higher similarity in their prototype will gather in some lighter gray area. Therefore, we expect the input patterns with higher similarity will be projected into some lighter gray area.

Via the U-Matrix, we can investigate the hierarchical relations between the trained units and of course the original data instances

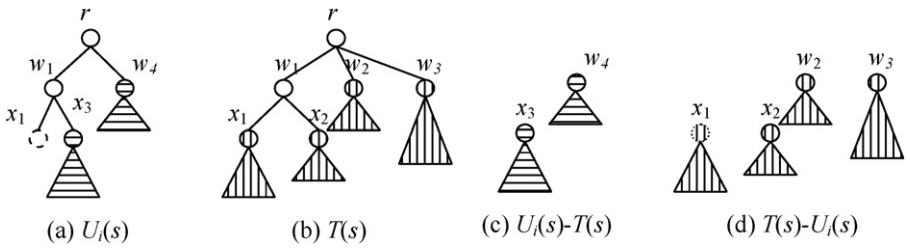


Fig. 9. The subtrees in the negative and the positive part of a concept tree $U_i(s)$ when an input pattern $T(s)$ is given. (a) $U_i(s)$ and the growing points, r and w_1. (b) $T(s)$. (c) The subtrees in the negative part. (d) The subtrees in the positive part. Since the node x_1 in $U_i(s)$ is partial grown, x_1 is only partially in $T(s)$–$U_i(s)$.

PrototypeAdaptation(T, BMU, s)
// **Input:** Input pattern T, BMU, training time s;
// **Output:** Adapted prototypes of the neighboring neurons around the BMU;
Decrease $\alpha(s)$ monotonically with s;
Decrease $\sigma(s)$ monotonically with s;
Let (x, y) be the location of the BMU on the map;
Let $N(x, y)$ be the neighborhood centered at (x, y) within a specified range, $\sigma(s)$;
For each neuron i in $N(x, y)$
  Let *positive* link the subtrees in T-U_i;
  Let *growPoints* link the nodes in U_i that new branches will grow;
  Let *negative* link the subtrees in U_i -T;
  Let (u,v) be the location of i on the map;
  Let r= dist((u, v), (x, y)); // Euclidean distance
  Let h=α*exp(-(r*r)/(2*σ*σ));
  Let $h_1 = h_2 = h$;
  U_i.ManySubtreesGrow(*growingPoints*, *positive*, h_1);
  U_i.ManySubtreesPrune(*negative*, h_2);

Fig. 10. Algorithm of prototype adaptation.

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

Fig. 11. Algorithm of ManySubtreesGrow.

easily. Another ways to present the hierarchical relations between the trained units on the map are to use the SOM models which have hierarchical maps, like the hierarchical feature map (HFP) [25] and the growing hierarchical self-organizing map (GHSOM) [26]. Those SOM variants actually create hierarchical relations between the trained units but they are complicated. The U-Matrix is simple.

After displaying the U-Matrix, we draw circles on the 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 neural unit or the neighboring units.

5. Experiments

We conducted five experiments to evaluate the abilities of the SOMCD. The first experiment was a robustness testing of the SOMCD. The second demonstrated the mapping quality of the SOMCD. We observed the topological preservation and the visualization ability of the SOMCD and the effect of the concept hierarchy types on the SOMCD. The third demonstrated the application of the SOMCD on real transactional data. In the third and the fourth experiments, we observed the mapping and clustering quality of the SOMCD on real categorical data and mixed data, respectively. The clustering results of the SOMCD were compared with state-of-the-art clustering algorithms and extended SOM models. The SOMCD, SOM, and SCM were all implemented in JAVA on a PC with an Intel's Core 2 CPU 6420 (2.13 GHz) and 4 GB memory running Linux 2.2.67.

In the first three experiments, the SOMCD and two comparing models, the SCM and the traditional SOM, were performed. The neurons of these three SOM models were all arranged in rectangular lattices on 2-D maps. The parameters setting of the SOMCD and the SOM referred to the suggestions of the SOM_PAK [22] and the SOM_Toolbox [21]. The number of the neurons on a map was about $5\sqrt{N}$, where N is the number of data instances in a dataset.

In the SOMCD and the SOM, the Gaussian function was used in the neighborhood function. The learning rate was 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 was 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 was set

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

Fig. 12. Algorithm of ManySubtreesPrune.

to 10 times of the number of neurons. In the SCM, the Mexican hat was used as the neighborhood function according to the author's suggestion [6]. The Mexican hat is defined as follows:

$$h(i) = (1 - a \cdot b \cdot i^2) \cdot \exp(-a \cdot i^2) \quad (25)$$

where $a = 0.35$ is a constant, determining the width of Mexican hat and $b = 2.0 \times s/S$ varies with time s. The learning rate is $\alpha(s) = 0.25 \times S/(0.5 \times S + s)$, and the value of the threshold is $x(s) = 0.2 \times s/S$.

When executing the SCM and the SOM, we used type-0 concept hierarchies only. Data instances were needed to convert to binary vectors and weighted symbol strings for the SOM and the SCM, respectively. For example, in transactional data, if a transaction has only the 5th, 9th, and 11th items which come from a type-0 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. Before the experimental results, some measures for evaluating mapping and clustering quality are provided.

5.1. Measures for evaluating mapping and clustering quality

5.1.1. Mapping quality

Two measures were used to observe the mapping qualities of the comparing SOM models: 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 [27–29]. 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:

$$\text{Trustworthiness}(k) = 1 - \frac{2}{Nk(2N - 3k - 1)} \sum_{i=1}^{N} \sum_{j \in U_k(i)} (r(i,j) - k) \quad (26)$$

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 [27–29]. The measure of the continuity of a mapping is calculated as follows:

$$\text{Continuity}(k) = 1 - \frac{2}{Nk(2N - 3k - 1)} \sum_{i=1}^{N} \sum_{j \in V_k(i)} (s(i,j) - k) \quad (27)$$

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.

5.1.2. Clustering quality

We evaluated the clustering quality of the comparing SOM models or the clustering algorithms by the weighted entropy and the purity. The first measure for clustering quality is the weighted entropy [18,30,31] defined as follows:

$$\text{Weighted entropy} = \sum_j \left(\frac{N_j}{N} \sum_i \left(-\frac{N_{ij}}{N_j} \cdot \log \frac{N_{ij}}{N_j} \right) \right) \quad (28)$$

where 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. The smaller weighted entropy is better in clustering quality. The second measure for clustering quality is the purity [17,19] defined as follows:

$$\text{Purity} = \sum_j \max_i \frac{N_{ij}}{N} \quad (29)$$

The purity is the accuracy, and the error rate [10,32] is 1 – purity. The higher purity is better in clustering quality.

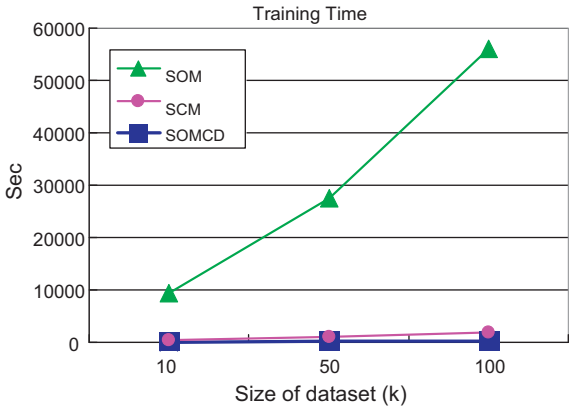


Fig. 13. The comparison of SOMCD, SCM and SOM in training time.

5.2. Robustness testing

When users deal with transactional data, the amount of the transactions may be very tremendous in many real cases. In this experiment, we tested the robustness of the SOMCD on the transactional data. Before we conducted this experiment, a virtual retailer which has 100,000 products (items) was created. Suppose all of the products form a three-level 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, were created, respectively. All of the transactions were 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 were 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.

We compared the execution time of the SOMCD and the two comparing SOM models in the training phase with the above three datasets; the result is depicted in Fig. 13. It is obvious that the SOMCD outperformed both of the SCM and the SOM.

5.3. Experiments on transactional datasets

5.3.1. Synthetic dataset: the simulative library checkouts

In the following experiment, we observed the visualization and the mapping qualities of the SOMCD and compared it with the SCM and the traditional SOM. Besides, we investigated whether the different types of the concept hierarchy affect the mapping result of the SOMCD. A simulative college library which has 1250 books was 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 the books belonging to PH.1 subcategory.

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

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 were further divided into three and two subclasses, as shown in Tables 2 and 3, respectively. Totally, there are seven groups distributed differently in the library dataset.

Each checkout of different classes was 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 was set to nine. That is, in this simulative library, the average number of books borrowed by students in each checkout was ten. Whenever the size of a checkout was determined, the books in a checkout were decided by a checkout content distribution of some class or subclass in Tables 1, 2 or 3. 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.

Fig. 14(a) and (b) shows the results of the library dataset mapped by the SOMCD when we used the type-1 and type-0 concept hierarchies, respectively. We distinguished these two situations of the SOMCD by SOMCD-1 and SOMCD-0. Fig. 14(c) and (d) shows the mapping results of the two comparing models, the SCM and the SOM, respectively. Each map was displayed by a 15 by 15 U-Matrix as its background. After displaying the U-Matrix, we drew circles on the units with nonzero BMU count in foreground. The area of each circle depended on the unit's BMU count.

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

In order to compare the mapping quality in visualization, for each neuron with non-zero BMU count, we randomly scattered the input patterns with specific colors within the neuron's circle. The colors of the input patterns were based on the classes to which they belong. Seven colors were assigned to the classes and the subclasses. In class I, the three subclasses of the checkouts were colored by red, green, and yellow, respectively. In class II, the two subclasses of the checkouts were colored by blue and magenta, respectively. The checkouts in class III and IV were colored by cyan and orange, respectively.

It is obvious the SOMCD-1 had a better visualization and mapping quality than the others. In Fig. 14(a), the boundaries are clear enough to separate the checkouts into different clusters easily. The checkouts which belong to the same class were gathered as a group or distributed as nearer neighbors, and the structure of the U-Matrix perfectly coincided with the structure of our testing dataset. When comparing with the other SOM models, we had the SOMCD-1 gathered the three subclasses of class I and the two subclasses of class II closely than the other classes. The SOMCD-1 reflected the actual structure of the dataset on the map more precisely than the others.

The result of the SOMCD-0 is shown in Fig. 14(b). The SOMCD-0 did not perform well as the SOMCD-1, but most the neurons with the same subclass or class gathered and distributed as nearer neighbors like the SOMCD-1 and the boundaries in U-Matrix could still be observed. From this result, it is clear that the relevancy information among the items in a concept hierarchy actually affected the mapping quality of the SOMCD.

In Fig. 14(c), not only are the boundaries of its U-Matrix not clear but also the SCM did not exactly map the same class of the checkouts together. For instance, class IV was split into two clusters located in the upper-left and the center of the map. In Fig. 14(d), the SOM had the similar problems as the SCM, and the checkouts tended to be projected all over the neurons on the map.

Fig. 15 shows the trustworthiness and the continuity values versus the number of neighbors, k, varying from 50 to 500, when the simulative library dataset was mapped by the three comparing models. It is obvious the SOMCD-1 outperformed the others. All the values of the trustworthiness and the continuity of the SOMCD-1 were greater than 0.80 and better than those of the others. That is, the neighborhood preservation and the mapping quality of the SOMCD-1 were better than those of the others. The SOMCD-0 had better mapping quality than the SOM but less than the SCM.

5.3.2. 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" [33] 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, 3357 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 considered as 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.

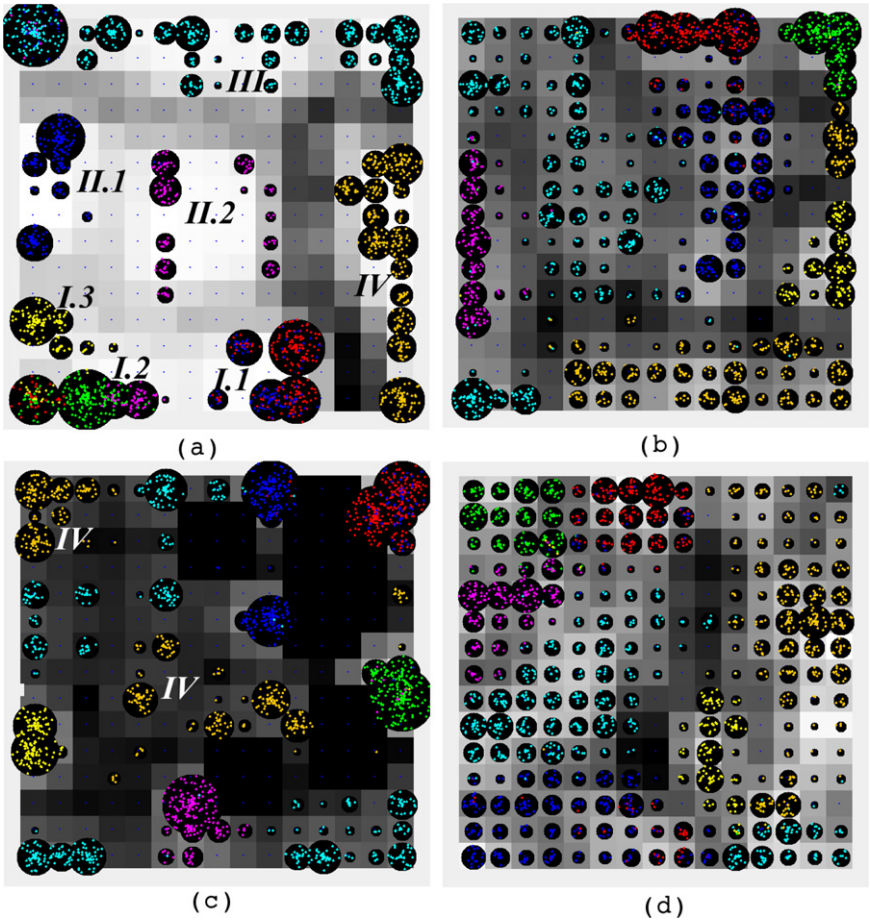


Fig. 14. The mapping results of the SOMCD, the SCM, and the SOM when using the simulative library dataset. (a) The mapping result of the SOMCD-1 (using type-1 C.H.). (b) The result of the SOMCD-0 (using type-0 C.H.). (c) The result of the SCM. (d) The result of the SOM. (For interpretation of the references to color in text, the reader is referred to the web version of the article.)

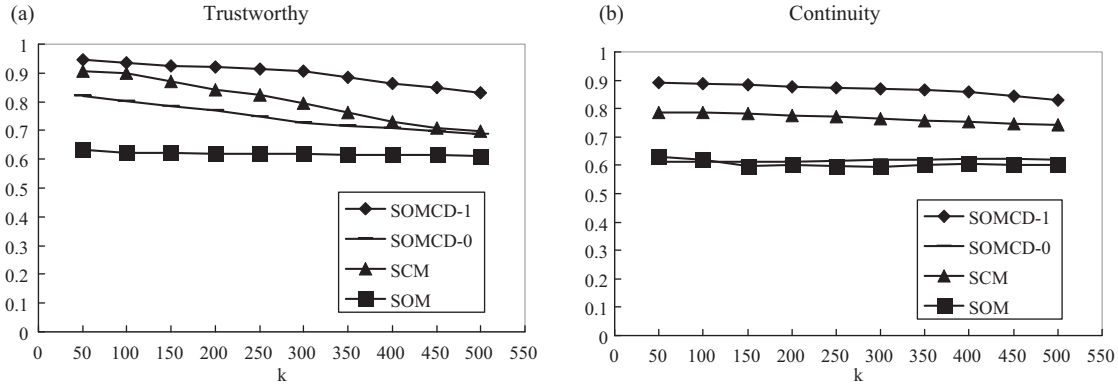


Fig. 15. Mapping qualities of the comparing models when using the simulative library dataset as an input. (a) The trustworthiness and (b) the continuity.

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, SOMCD, SCM, and SOM, are depicted in Fig. 16(a)–(c), respectively. Each map had 18 by 18 units and a U-Matrix displayed in the background. The boundaries in the SOMCD were clearer than the other two. With the aid of the U-Matrix, we could separate the units into different groups easily in the SOMCD, although there were many small units scattering over the map.

For validating and comparing purpose, each article was mapped into its BMU with a specific color. Five different colors, red, green, yellow, blue, and magenta, were assigned to SIGIR, SIGMOD, SIGCOMM, SIGCHI, and SIGMM, respectively. Contrast to the other

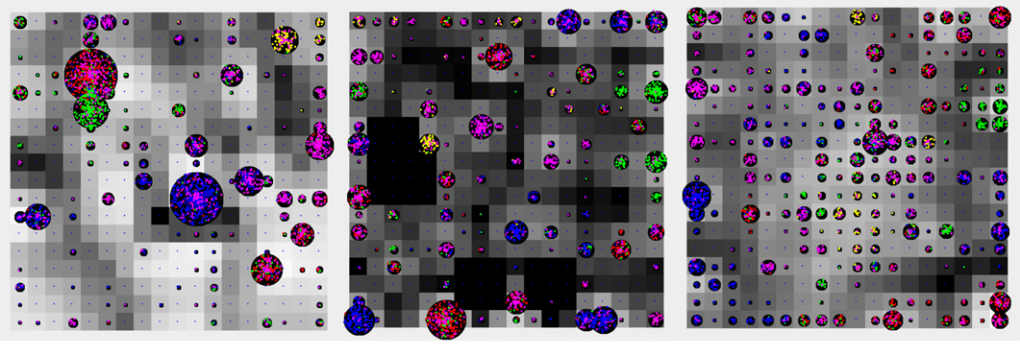


Fig. 16. The mapping results of the ACM proceedings' articles: (a) is of the SOMCD, (b) is of the SCM, and (c) is of the SOM. Each map has 18 by 18 units. (For interpretation of the references to color in text, the reader is referred to the web version of the article.)

models, the SOMCD had the following advantages in mapping and interesting findings. (1) The SOMCD projected a large amount articles of the same proceeding into few units; (2) the topics of the SIGIR and the SIGMOD are closer than the other proceedings, since the red and green units were close; (3) the topics in the SIGCOMM are more concentrated than the other proceedings, since the yellow units were only distributed on the upper right corner; (4) the topics in the SIGMM are more diverse than the other proceedings and overlap the topics in the other proceedings in some extent, since quite a few magenta points were mixed with the other color points in many units.

In order to confirm the above findings, we did more analyses as follows. Firstly, we removed the articles of the SIGMM in the dataset and applied the SOMCD on the remained dataset. We had the resulting map depicted in Fig. 17 was cleaner and clearer than the map in Fig. 16(a). Different color points were almost mapped into different units, and the purity of each unit was high. It is reasonable for us to confirm the last finding.

Next, articles mapped into the following four neurons, (4, 3), (10, 10), (4, 5), (15, 1) in Fig. 16(a), were analyzed, since they had the largest number of articles in the respective proceedings, SIGIR, SIGCHI, SIGMOD, and SIGCOMM. There were about 51.8% of the SIGIR's articles in neuron (4, 3), 47.6% of the SIGCHI's articles in neuron (10, 10), 34.5% of the SIGMOD's articles in neuron (4, 5), and 67.7% of the SIGCOMM's articles in neuron (15, 1). Besides, the first three neurons were the top three in BMU count; there were about 16.7%, 16.6%, 7.0%, and 4.3% of the articles mapped into the four neurons, respectively. Note that (0, 0) was located at the upper left corner on the map.

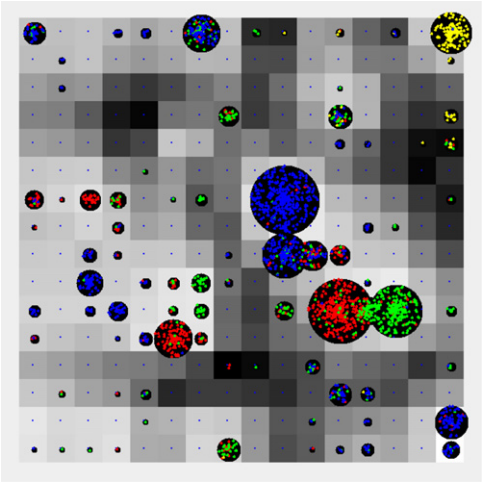


Fig. 17. The mapping results of the ACM proceedings' articles by the SOMCD, when the articles of the SIGMM were removed. (For interpretation of the references to color in text, the reader is referred to the web version of the article.)

For each of the four 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. 18(a)–(d) depicts the frequency distributions of the four 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, were 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 showed the different proceedings.

In Fig. 18, it is clear that the four neurons had different kinds of distributions in their classification codes. However, the neurons (4, 3) and (4, 5) contained quite a few common classification codes in H.3, and the classification codes in neuron (15, 1) were totally different from the other three neurons. These helped us confirm that some topics in SIGIR and SIGMOD are close and the SIGCOMM's topics are farer from the others. Also, for each of the four neurons in Fig. 18(a)–(d), we found the corresponding proceeding's articles actually had the common classification codes with the SIGMM's. This helped us validate the last finding again.

Besides, we found in Fig. 18(a)–(d) most of the classification codes concentrated on one level-2 classification code, that is, the articles inside each of the four neurons must have high relevancy. We inspected the articles inside neuron (4, 3), we found "information search, retrieval, indexing" was the main topic of the articles inside this neuron. We found even those of articles not coming from the SIGIR were also relevant to information retrieval. For instance, the article "Comprehensive query-dependent fusion using regression-on-folksonomies: a case study of multimodal music search" [33], which come from SIGMM, is obviously related to multimedia as well as information search and retrieval. Thus, it is reasonable this article was projected into this neuron. This helped us validate the mapping result of our model again.

In the end of this subsection, we compared the three SOM models in training time. From the real world dataset, the ACM proceedings' articles, we chose articles randomly and created three datasets of which size were 1357, 2357, and 3357. Then we applied the SOMCD, the SOM and the SCM, to those three datasets, respectively and compared their training time. The result was depicted in Fig. 19. We found that our model was much faster than the other two models. The SOM performed worst; since each transaction in the SOM must be converted to a binary vector of which dimension is the number of the all possible items in a transactional dataset. In the case of ACM proceedings' articles, it is about 1215 items. The SOM suffered from the high dimensional problem severely, while

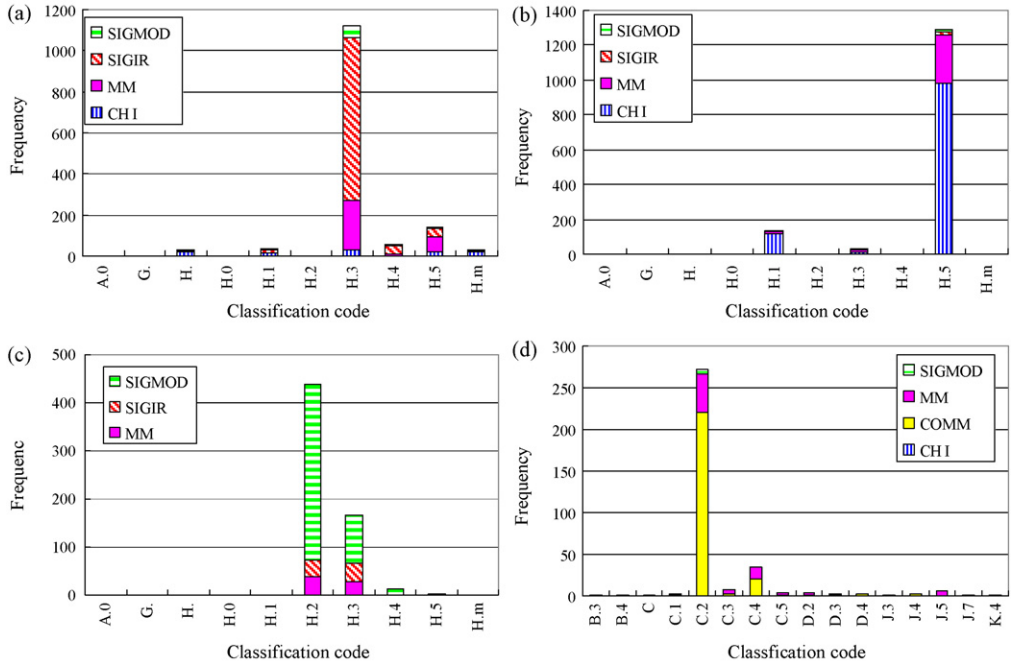


Fig. 18. The frequency distributions of the classification codes of the selected neurons. (a), (b), (c), (d) are the histograms of the neurons (4, 3), (10, 10), (4, 5), (15, 1), respectively.

the SOMCD did not have such a problem. Transactions in the SOMCD are represented as trees but not vectors.

5.4. Experiment on categorical datasets

Owing to the lack of type-1 categorical data sets, we only conducted experiments on type-0 categorical datasets. Two type-0 categorical data sets, the Mushroom and the Congressional Voting Records, which come from the UCI machine learning repository [34], were used to evaluate the quality of the SOMCD in mapping and clustering categorical datasets.

The Mushroom dataset has 8124 instances, and each instance has 22 categorical attributes plus a class label. Those attributes describe the physical characteristics of Mushrooms and are all nominally valued. All of the 8124 Mushroom instances are labeled by either edible or poisonous. 4208 of them are edible (51.8%) and 3916 are poisonous (48.2%).

The Congressional Voting Records dataset is the United States Congressional voting records in 1984. There are 435 records, and each record has 16 attributes plus a class label. Each record reveals the votes on 16 issues of a Congressman in 1984 and is labeled by the Congressman's party, either Republican or Democrat. 168 of them are Republicans and 267 are Democrats. Missing values occur in both of the two datasets.

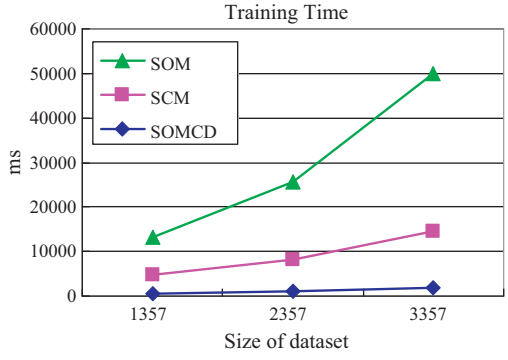


Fig. 19. The training time comparison of the SOMCD, SOM, SCM on the ACM proceedings' articles.

The Mushroom and the Congressional Voting Records datasets had been used for evaluating clustering performance in many transactional and categorical data clustering algorithms, including Large Items [16], ROCK [15], CLOPE [17], COOLCAT [31], TCSOM [10], CLICKS [35], AT-DC [32], SCALE [18,19] and so on. Except COOLCAT and SCALE, all the algorithms used both of the datasets. The results of the SOMCD on the two datasets were compared with the mentioned algorithms.

In Fig. 20, we show the results of the Mushroom dataset mapped by the SOMCD. The map in Fig. 20(a) displays the neurons with non-zero BMU count in the foreground and the U-Matrix in the background; and the data instances colored by either red (poisonous) or green (edible) were mapped into their own BMUs. The colored instances were used for observing the mapping quality only. We found all the neurons were pure except only two neurons; and neurons of the same class were gathered as a group. The boundaries in the U-Matrix were clear, so we could easily and manually divide the neurons with non-zero BMU count into 16 clusters from the map in Fig. 20(b). The clusters are shown in Table 5. Only three clusters were impure. If we clustered the neurons finer, some clusters, like the cluster 1, 5, 6, D, could be further divided into smaller clusters. Totally, 26 clusters were obtained. We evaluated the clustering quality by the weighted entropy and the purity. The clustering quality of the SOMCD and other algorithms on the Mushroom dataset is listed in Table 6. We found the clustering quality of the SOMCD was close to quality of the CLOPE and ROCK and is better than the quality of the others.

In Fig. 21, we show the results of the Congressional Voting Records mapped by the SOMCD. The map in Fig. 21(a) displays the neurons with non-zero BMU count in the foreground and the U-Matrix in the background; and the data instances colored by either red (Republicans) or green (Democrat) were mapped into their BMU. The boundaries in the U-Matrix were clear, so we could easily and manually divide the neurons with non-zero BMU count into five clusters from the map in Fig. 21(b). The clustering result is

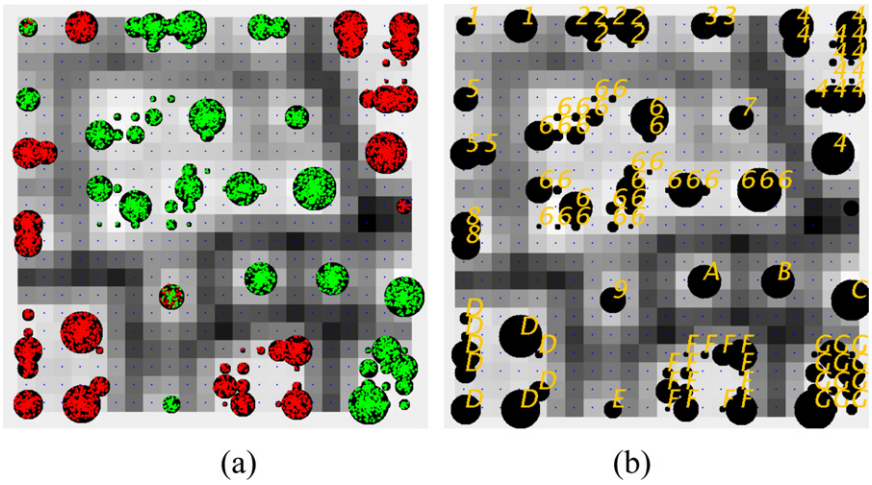


Fig. 20. The mapping and clustering results of the SOMCD on the Mushroom. (a) The map displays the neurons with non-zero BMU count and the U-Matrix in the background; and the data instances colored by red (poisonous) and green (edible) were mapped into their own BMUs. (b) The manual clustering result. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

Table 5
The clusters of the Mushroom on the SOMCD's map.

Cluster	Edible	Poisonous
1	48	200
2	512	0
3	192	0
4	0	1332
5	96	264
6	1735	0
7	96	0
8	0	288
9	41	72
A	192	0
B	192	0
C	288	0
D	0	1053
E	48	0
F	0	707
G	768	0

Table 6
The clustering quality of the SOMCD and the mentioned algorithms on the Mushroom dataset.[a]

Algorithm	No. of Clusters	Entropy	Purity
SOMCD	16	0.0719	0.9772
SOMCD	26	0.0172	0.9940
LargeItems	14	0.1745	0.9628
ROCK	21	0.0011	0.9961
CLOPE	27 (r = 2.6)	n/a	0.9961
CLOPE	30 (r = 3.1)	0.0000	1.0000
TCSOM	2–9	n/a	0.8180
AT-DC	8	0.2350	0.9321
CLICKS	11	n/a	0.8843

[a] Entropy of the original dataset = 0.9991.

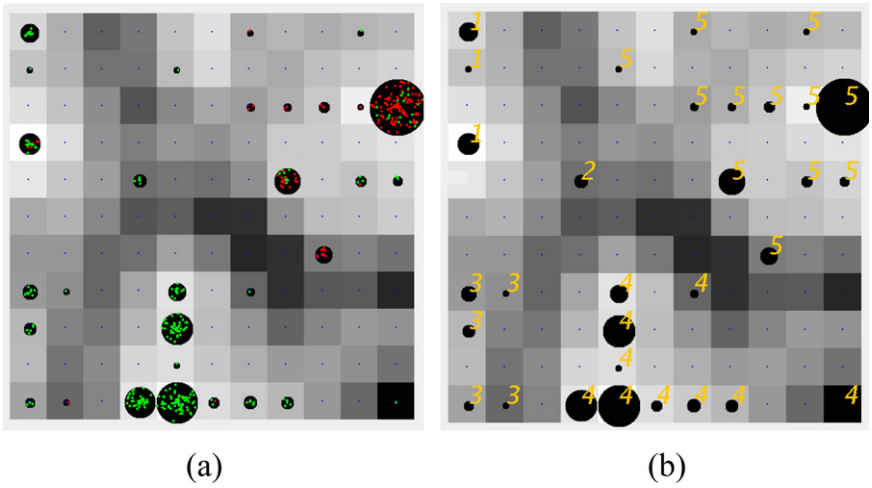


Fig. 21. The mapping and clustering results of the SOMCD on the Congressional Voting Records. (a) The map displays the neurons with non-zero BMU count and the U-Matrix in the background; and the data instances colored by red (Republican) and green (Democrat) were mapped into their own BMUs. (b) The manual clustering result. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

Table 7
The clusters of the Congressional Voting Records on the SOMCD's map.

Cluster no.	Democrat	Republican
1	26	6
2	6	0
3	17	1
4	180	4
5	38	157

shown in Table 7. Since the color of the boundaries among cluster 1, 2, 3, and 4 was lighter than the color of the boundary around cluster 5, cluster 1, 2, 3, and 4 could be merged into one cluster. The result coincided with the structure of our dataset, which contains only two classes of instances. The clustering quality of the SOMCD and other algorithms on the Congressional Voting Records is listed in Table 8. We found the clustering quality of the SOMCD was better than the other algorithms except the TCSOM.

5.5. Experiment on mixed dataset

The Adult dataset, which comes from the UCI machine learning repository [34], was used to evaluate the quality of the SOMCD in mapping and clustering mixed data. The Adult dataset has 48,842 instances, and each instance has 14 attributes, including eight categorical attributes and 6 numerical attributes, plus one class label. The class label indicates whether the salary is over 50k or not. It is about 24% of the instances having the label '>50k' and 76% having the label '≤50k'.

The Adult dataset had been used for evaluating the clustering performance of some extended SOM models and clustering algorithms on mixed data, including GSOM [12], GViSOM [14], and CAVE [30]. In the experiments of the GSOM and the GViSOM, instead of using the whole Adult dataset, the dataset has only 10,000 instances drawn randomly from the original Adult dataset, and for each instance, only seven attributes were selected. There are four numerical ones, Capital_gain, Capital_loss, Age, and Hours_per_week; and three categorical ones, Marital_status, Relationship, and Education. This data subset has about 24.24% instances having the label '>50k' and 75.76% instances having the label '≤50k', similar to the distribution of the original Adult dataset. In order to compare with the mentioned SOM models, we adopted the same data subset in our experiment.

Next, we built the accompanying concept hierarchy for the Adult subset. For each numerical attributes, we created a category node and a virtual node as its only one leaf. For each categorical attributes, we adopted the same concept hierarchies in the mentioned SOM models as shown in Fig. 22. The Marital-status and the Education attributes are multi-level concept hierarchies, which usually contain more information about the relevancy between their own items. Each data instance in the Adult subset was converted to a transaction tree as we have mentioned in Section 3.4. The CDF used in this experiment is the uniform distribution function.

Table 8
The clustering quality of the SOMCD and the mentioned algorithms on the Congressional Voting Records.[a]

Algorithm	No. of clusters	Entropy	Purity
SOMCD	5	0.4469	0.8874
SOMCD	2	0.4681	0.8894
TCSOM	2–9	n/a	0.9210
ROCK	2	0.5360	0.8571
AT-DC	2	0.5402	0.8111
CLICKS	6	n/a	0.7954
LargeItems	2	0.5360	0.8571

[a] Entropy of the original dataset = 0.962308.

Table 9
The clusters of the Adult on the SOMCD's map.

Cluster	≤50k	>50k	>50k (%)
B	97	346	78.10
A	266	534	66.75
6	118	180	60.40
8	155	135	46.55
7	431	340	44.10
F	176	80	31.25
5	129	56	30.27
D	1094	452	29.24
C	220	35	13.73
H	293	43	12.80
3	444	50	10.12
E	884	98	9.98
9	933	39	4.01
2	746	19	2.48
G	126	3	2.33
4	532	9	1.66
I	82	1	1.20
1	713	4	0.56
J	137	0	0.00
Total	7576	2424	

In Fig. 23, we show the results of the Adult data subset mapped by the SOMCD. The map in Fig. 23(a) displays the neurons with non-zero BMU count in the foreground and the U-Matrix in the background; and the data instances colored by either red ('≤50k') or green ('>50k') were mapped into their own BMUs. We found most of the neurons had high purity. The boundaries in the U-Matrix were clear, so we could easily and manually divide the neurons with non-zero BMU count into 19 clusters from the map in Fig. 23(b). Table 9 shows the clustering results which were sorted according to the percentage of the '>50k' instances in each cluster. Most of the percentages were significantly different from the average value 24.24%. If we used the darker units as the boundaries, then we obtained the larger regions surrounded by the darker boundary and the number of the clusters was reduced to 7. Each of the following cluster groups, (1, 2, 3, 4), (5, 6), (7, 8, A), (9, J), (C, D), (E, F, H, I, G), were merged as a larger cluster, respectively.

In order to know whether the relevancy information in concept hierarchies was useful for the mapping and clustering quality of the SOMCD, we further conducted the experiment with the three categorical attributes having only type-0 concept hierarchies. Like the experiments on the library dataset, we distinguished these two situations (using type-1 or type-0 concept hierarchies) of the SOMCD by SOMCD-1 and SOMCD-0, respectively.

In Fig. 24, we show the results of the Adult data subset mapped by the SOMCD-0. We found the mapping result was close to the result of SOMCD-1. Next, we also clustered the neurons with non-zero BMU count on the map in Fig. 23(b) manually and obtained 15 clusters.

The clustering quality of the SOMCD and the other two SOM models and the CAVE on the Adult dataset is listed in Table 10. We found (a) the clustering quality of the SOMCD-1 is better than the

Table 10
Clustering quality of SOMCD, GSOM, GViSOM and CAVE on the Adult dataset.[a]

Algorithm	No. of clusters	Entropy	Purity
SOMCD-1	19	0.5672	0.8155
SOMCD-1	7	0.5927	0.7982
SOMCD-0	15	0.6110	0.7920
GSOM	7 (212 outliers)	0.6225	0.7715
GViSOM	5 (6 outliers)	0.61773	0.7807
CAVE[b]	9	0.6069	0.7871

[a] Entropy of the Adult data subset is 0.7990.
[b] The CAVE used the whole Adult dataset.

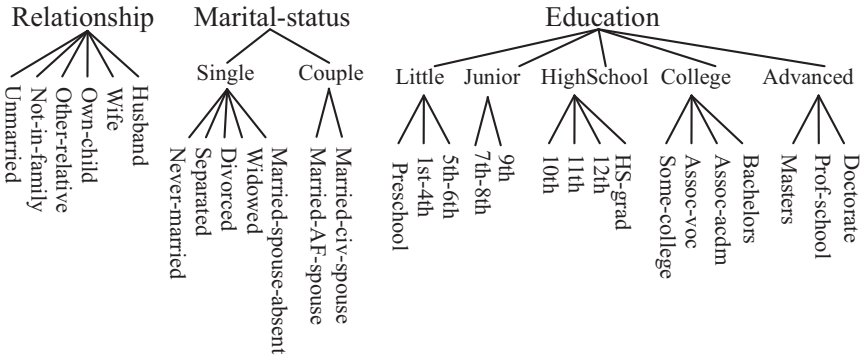


Fig. 22. The concept hierarchies of the three categorical attributes in the Adult dataset.

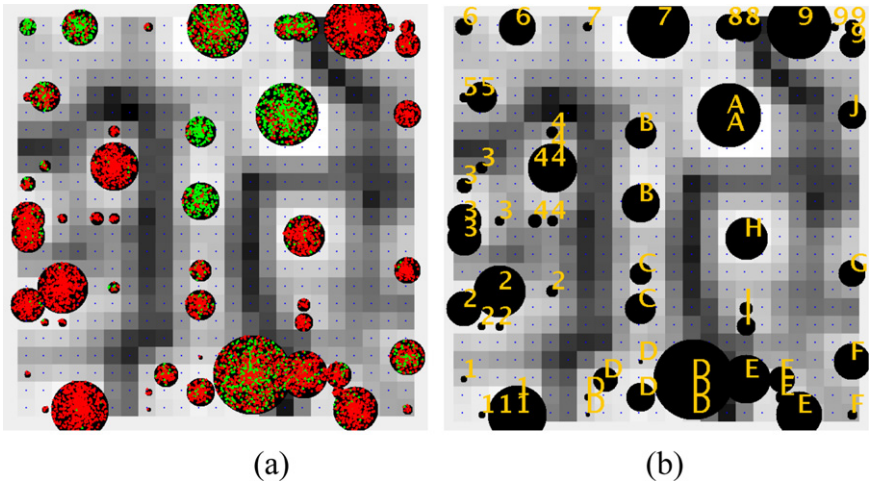


Fig. 23. The mapping and clustering results of the SOMCD on the Adult. (a) The map displays the neurons with non-zero BMU count and the U-Matrix in the background. The data instances colored by red ('≤50k') and green ('>50k') were mapped into their own BMUs. (b) The clustering result. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

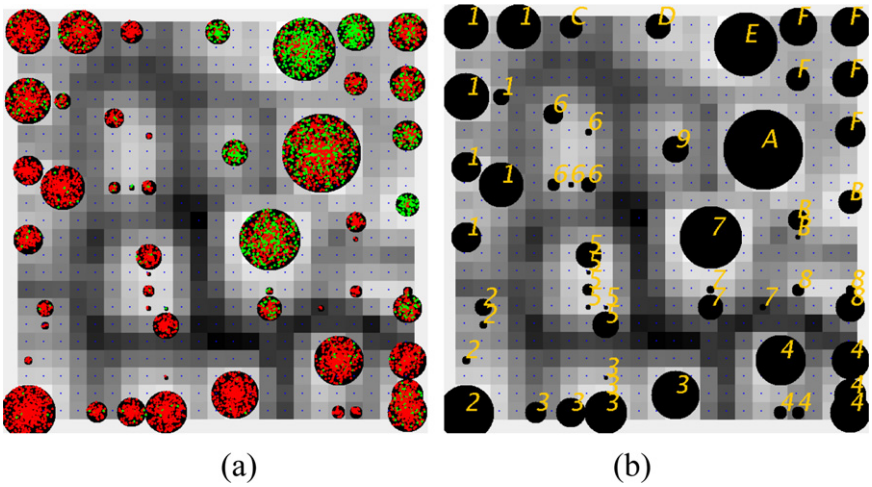


Fig. 24. The mapping and clustering results of the SOMCD-0 on the Adult. (a) The map displays the neurons with non-zero BMU count and the U-Matrix in the background. The data instances colored by red ('≤50k') and green ('>50k') were mapped into their own BMUs. (b) The clustering result. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

other two SOM models even we clustered the neurons in a rough manner; (b) the clustering performance of the SOMCD-0 is as good as the other SOM models but is worse than SOMCD-1. From the above findings, the relevancy information embedded in the concept hierarchies actually helped the SOMCD gain better performance.

6. Conclusions and future work

In this paper, we propose an extended SOM model, the SOMCD, which can map the varied kinds of data objects all related to the categorical domain into a lower dimensional space and visualize

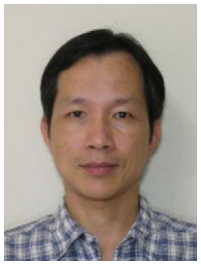
the topological order of the data objects. Based on the concept trees to represent the data objects in the categorical domain and the prototypes of the neurons, the new devised distance measure can take the relevancy information embedded in the accompanying concept hierarchies into account and properly find the similarity between the data objects and the neurons. After adapting the neurons' tree-structure prototypes iteratively, the SOMCD has the good qualities in projecting and visualization. Users of the SOMCD can easily separate the trained neurons on the map into different groups hierarchically and cluster the original data objects eventually. From the experiments in the synthetic and real datasets, the SOMCD performs better than many other SOM models and clustering algorithms in visualization, mapping and clustering. All the good natures of the SOMCD mainly come from the tree-structure prototype, the new devised distance measure, and the tree-growing adaptation method. Our future work is to develop a clustering algorithm for the trained SOMCD neurons in order to automatically and precisely cluster the varied kinds of data objects which are in the categorical domain.

Acknowledgement

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

References

[1] T. Kohonen, Self-organized formation of topologically correct feature maps, Biological Cybernetics 43 (1982) 59–69.
[2] T. Kohonen, Self-organizing Maps, 3rd ed., Springer, Berlin, 2001.
[3] T. Kohonen, P. Somervuo, Self-organizing maps of symbol strings, Neurocomputing 21 (1998) 19–30.
[4] T. Kohonen, P. Somervuo, How to make large self-organizing maps for nonvectorial data, Neural Networks 15 (2002) 945–952.
[5] T. Kohonen, Self-organizing maps of symbol strings, in: Technical Report A42, Laboratory of Computer and Information Science, Helsinki University of Technology, Finland, 1996.
[6] J.A. Flanagan, Unsupervised clustering of symbol strings, in: Proceedings of the International Joint Conference on Neural Networks, 2003, vol. 3254, 2003, pp. 3250–3255.
[7] J. Himberg, J.A. Flanagan, J. Fantyjarvi, Towards context awareness using Symbol Clustering Map, in: Proceedings of WSOM, 2003, Kitakyushu, Japan, 2003.
[8] P.J. Somervuo, Online algorithm for the self-organizing map of symbol strings, Neural Networks 17 (2004) 1231–1239.
[9] S. Günter, H. Bunke, Self-organizing map for clustering in the graph domain, Pattern Recognition Letters 23 (2002) 405–417.
[10] Z.Y. He, X.F. Xu, S.C. Deng, TCSOM: clustering transactions using self-organizing map, Neural Processing Letters 22 (2005) 249–262.
[11] B. Hammer, A. Micheli, N. Neubauer, A. Sperduti, M. Strickert, Self-organizing maps for time series, in: Proceedings of WSOM, 2005, Paris, 2005.
[12] C.-C. Hsu, Generalizing self-organizing map for categorical data, IEEE Transactions on Neural Networks 17 (2006) 294–304.
[13] J. Han, M. Kamber, Data Mining: Concepts And Techniques, Morgan Kaufmann, San Francisco, CA, London, 2001.
[14] C.C. Hsu, K.M. Wang, S.H. Wang, GViSOM for multivariate mixed data projection and structure visualization, in: Proceedings of the International Joint Conference on Neural Networks, 2006, 2006, pp. 3300–3305.
[15] S. Guha, R. Rastogi, K. Shim, Rock: a robust clustering algorithm for categorical attributes, Information Systems 25 (2000) 345–366.
[16] K. Wang, C. Xu, B. Liu, Clustering transactions using large items, in: Proceedings of the Eighth International Conference on Information and Knowledge Management, ACM, Kansas City, MI, United States, 1999, pp. 483–490.
[17] Y. Yang, X. Guan, J. You, CLOPE: a fast and effective clustering algorithm for transactional data, in: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, Edmonton, Alberta, Canada, 2002, pp. 682–687.
[18] H. Yan, K. Chen, L. Liu, Z. Yi, SCALE: a scalable framework for efficiently clustering transactional data, Data Mining and Knowledge Discovery 20 (2010) 1–27.
[19] H. Yan, K. Chen, L. Liu, J. Bae, Determining the best K for clustering transactional datasets: a coverage density-based approach, Data & Knowledge Engineering 68 (2009) 28–48.
[20] H. Yin, ViSOM—a novel method for multivariate data projection and structure visualization, IEEE Transactions on Neural Networks 13 (2002) 237–243.
[21] J. Vesanto, J. Himberg, E. Alhoniemi, J. Parhankangas, SOM toolbox for Matlab 5, in: Report A57, Helsinki University of Technology, 2000, Available from http://www.cis.hut.fi/projects/somtoolbox/.
[22] T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, SOM_PAK. The self-organizing map program package, in: Report A31, Helsinki University of Technology, 1996, Available from http://www.cis.hut.fi/research/som_pak/.
[23] A. Ultsch, Self-organizing neural networks for visualization and classification, in: O. Opitz, B. Lausen, R. Klar (Eds.), Information and Classification, Springer, 1993, pp. 307–313.
[24] A. Ultsch, Maps for the visualization of high-dimensional dataspaces, in: Proceedings of the WSOM, 2003, Kitakyushu, Japan, 2003, pp. 225–230.
[25] R. Mukkulainen, Script recognition with hierarchical feature maps, Connection Science 2 (1990) 83–101.
[26] A. Rauber, D. Merkl, M. Dittenbach, The growing hierarchical self-organizing map: exploratory analysis of high-dimensional data, IEEE Transactions on Neural Networks 13 (2002) 1331–1341.
[27] S. Kaski, J. Nikkila, M. Oja, J. Venna, P. Toronen, E. Castren, Trustworthiness and metrics in visualizing similarity of gene expression, BMC Bioinformatics 4 (2003) 48.
[28] A. Vathy-Fogarassy, A. Janos, Local and global mappings of topology representing networks, Information Sciences 179 (2009) 3791–3803.
[29] J. Venna, S. Kaski, Local multidimensional scaling with controlled tradeoff between trustworthiness and continuity, in: Proceedings of the Workshop on Self-Organizing Maps, 2005, 2005, pp. 695–702.
[30] C.-C. Hsu, Y.-C. Chen, Mining of mixed data with application to catalog marketing, Expert Systems with Applications 32 (2007) 12–23.
[31] D. Barbará, Y. Li, J. Couto, COOLCAT: an entropy-based algorithm for categorical clustering, in: Proceedings of the Eleventh International Conference on Information and Knowledge Management, ACM, McLean, VI, USA, 2002, pp. 582–589.
[32] E. Cesario, G. Manco, R. Ortale, Top-down parameter-free clustering of high-dimensional categorical data, IEEE Transactions on Knowledge and Data Engineering 19 (2007) 1607–1624.
[33] B. Zhang, Q. Xiang, H. Lu, J. Shen, Y. Wang, Comprehensive query-dependent fusion using regression-on-folksonomies: a case study of multimodal music search, in: Proceedings of the 17th ACM International Conference on Multimedia, ACM, Beijing, China, 2009, pp. 213–222.
[34] A. Frank, A. Asuncion, UCI Machine Learning Repository, University of California, School of Information and Computer Science, Irvine, CA, 2010, http://archive.ics.uci.edu/ml.
[35] M.J. Zaki, M. Peters, I. Assent, T. Seidl, Clicks: an effective algorithm for mining subspace clusters in categorical datasets, Data & Knowledge Engineering 60 (2007) 51–70.



Wen-Chung Liao received his M.S. degree from the Department of Applied Mathematics, National Chung Hsing University, Taiwan, in 1990. He is a doctoral student in the Department of Information Management, National Yunlin University of Science and Technology, Taiwan. His research interests include data mining, machine learning, and statistics.

Chung-Chian Hsu received the M.S. and Ph.D. degrees in computer science from Northwestern University, Evanston IL, USA, in 1988 and 1992, respectively. He joined the Department of Information Management at National Yunlin University of Science and Technology, Taiwan, in 1993. He was the Chairman of the Department from 2000 to 2003. He is currently a professor at the Department. Since 2002, he has also been the director of the Information Systems Division at the Testing Center for Technological and Vocational Education, Taiwan. His research interests include data mining, machine learning, pattern recognition, information retrieval, and decision support systems.