Knowl Inf Syst (2013) 37:251–278
DOI 10.1007/s10115-012-0501-1

REGULAR PAPER

A time-interval sequence classification method

Chieh-Yuan Tsai · Chih-Jung Chen · Chun-Ju Chien

Received: 11 May 2011 / Revised: 21 December 2011 / Accepted: 23 April 2012 /
Published online: 16 May 2012

Abstract Classification is one of the most popular behavior prediction tools in behavior informatics (behavior computing) to predict group membership for data instances. It has been greatly used to support customer relationship management (CRM) such as customer identification, one-to-one marketing, fraud detection, and lifetime value analysis. Although previous studies showed themselves efficient and accurate in certain CRM classification applications, most of them took demographic, RFM-type, or activity attributes as classification criteria and seldom took temporal relationship among these attributes into account. To bridge this gap, this study takes customer temporal behavior data, called time-interval sequences, as classification criteria and develops a two-stage classification framework. In the first stage, time-interval sequential patterns are discovered from customer temporal databases. Then, a time-interval sequence classifier optimized by the particle swam optimization (PSO) algorithm is developed to achieve high classification accuracy in the second stage. The experiment results indicate the proposed time-interval sequence classification framework is efficient and accurate to predict the class label of new customer temporal data.

Keywords Behavior informatics · Customer relationship management (CRM) · Classification · Time-interval sequences · Particle swarm optimization (PSO) algorithm

1 Introduction

With the increasing needs and focus on social network analysis and social computing, behavior informatics is now receiving much more attention [8,9]. Behavior informatics, also known as behavior computing, is defined as the scientific field that aims to develop methodologies,

C.-Y. Tsai (✉) · C.-J. Chen
Department of Industrial Engineering and Management, Yuan Ze University,
No. 135, Yuantung Rd., Chungli City, Taoyuan County 320, Taiwan
e-mail: cytsai@saturn.yzu.edu.tw

C.-J. Chien
Department of Supplier Quality Engineering, Showa Denko HD Trace Corp., Hsinchu, Taiwan

techniques, and practical tools for representing, modeling, analyzing, understanding and/or utilizing symbolic and/or mapped behavior, behavioral impacts, the formation of behavior-oriented groups and collective intelligence, and behavioral intelligence emergence [10]. Behavior informatics, which greatly complements behavioral science and behavior studies in social science, shows the potential for designing and supplying new and practical mechanisms, tools, and systems for deep behavior understanding and use. It can be widely used in many areas and domains, such as intrusion detection [54], social computing [55], behavior analysis of terrorist and criminals [11], and customer relationship management [50].

In customer relationship management (CRM), it is wildly agreed that customer behavior analysis is essential for deeply understanding customers, and eventually for boosting enterprise operation and enhancing business intelligence. CRM can be viewed as managerial efforts to manage business interactions with customers by combining business processes and technologies that seek to understand a company's customers [27]. Typically, the CRM framework can be classified into operational and analytical where operational CRM refers to the automation of business processes and analytical CRM refers to the analysis of customer characteristics and behaviors [47]. As such, analytical CRM can support the organization's customer management strategies and help an organization better discriminate and more effectively allocate resources to the most profitable group of customers [42].

To support the complicate tasks in the analytical CRM framework, classification is identified as one of the best supporting tools for CRM. Classification aims at building a model to predict future customer behaviors through classifying database records into a number of predefined classes. Examples of using classification techniques for CRM applications include direct marketing [16,53], customer identification [22,28], customer attraction [14,29], loyalty program [15,32], one-to-one marketing [24,30], fraud detection [57], and lifetime value analysis [3]. The popular classification algorithms in CRM include decision tree [28], artificial neural networks [22], logic regression [45], Bayesian network classifiers [4], genetic algorithms [1], and support vector machine [13]. A complete literature review of classification in CRM can be further found in the works of Lessmann and Voß [37] and Ngai et al. [42].

Although previous studies showed themselves efficient and accurate in certain CRM classification applications, most of them took demographic, RFM-type, or activity attributes as classification criteria and seldom took temporal relationship among these attributes into account [52]. For example, customer A has the behavior of "inkjet printer → ink cartridge → papers" and a customer B has the behavior of "papers → ink cartridge → inkjet printer." The two customers were regarded as having the same behavior of purchasing "inkjet printer", "ink cartridge", and "papers" by previous studies. However, it is inappropriate to make this claim since the two customers purchase products in totally different temporal order. In fact, not only the purchasing order should be taken into consideration, but also the time-interval information between purchasing activities should be included [60]. For example, customers C and D have the same purchasing behavior of "notebook → printer", but the time interval between the two purchasing activities for customer C is 1 week and the time interval between the two activities for customer D is 1 year. If the time-interval information between activities is not considered, the two customers would be classified as the same behavior group which is not acceptable in practice.

To solve the above difficulties, this research takes customer temporal behavior data, called time-interval sequences, as classification criteria. The time-interval sequence of a customer contains not only the temporal order of activities but also the time-interval information between them. An example of the time-interval sequence is "inkjet printer → (1 week) → papers → (3 months)→ ink cartridge." Based on the time-interval sequences, a two-stage classification framework is proposed. In the first stage, a sequential pattern mining technique, I-PrefixSpan algorithm, is applied to derive readable time-interval sequential patterns from customer behavior databases. In the second stage, a time-interval sequence classifier is developed to achieve the high classification accuracy while keeping users easily understand the inherent characteristics of the classification result. The rest of this paper is organized as follows. Section 2 introduces the background knowledge related to sequence classification. Section 3 introduces the details of the proposed classification framework. In Sect. 4, two cases are presented to show the computation process and parameter settings of the proposed framework. Finally, the conclusion and future works of this study are summarized in the last section.

2 Literature review

The sequence classification is to assign the most probable class label to a given sequence by a generative model (or classifier). Sequence classification problem arises in many real-world applications such as protein function prediction, text classification or speech recognition, but seldom found in CRM applications. Sequential data are sequences of ordered “events” or “activities”, where each event is described by a set of predicates. Examples of sequential data include text, bio-sequences (DNA, proteins), web-usage data, multiplayer games, plan-execution traces, etc. Xing et al. [59] introduce a novel concept of MPL (minimum prediction length) and develop ECTS (early classification on time series) method. ECTS makes early predictions and at the same time retains the accuracy comparable with that of a 1NN (1-nearest neighbor) classifier using the full-length time series. Köknar-Tezel and Latecki [31] presented an innovative approach that augments the minority class by adding synthetic points in distance space and use support vector machines for classification. The experimental results on standard time series show that synthetic points significantly improve the classification rate of the rare events, and in most cases also improve the overall accuracy of SVMs. Joung et al. [25] employed hidden Markov model (HMM), a statistical model in which the system being modeled is assumed to be a Markov process with unobserved state, to generate a sequence classification model. Bouchaffra and Tan [7] extended the traditional hidden Markov model named “structural hidden Markov model” by partitioning the set of observation sequences into classes of equivalences to enhance the accuracy for the sequence classification model. Bruyn et al. [6] integrated estimation, clustering, and classification into the traditional, three-step approach to make the result of sequence classification more reliable.

Recently, sequential pattern classification problems are treated as feature mining problem. The feature mining algorithm uses the extracted patterns as features. The patterns are vectorized based on their matched sequences and then standard classification method algorithms such as Naïve Bayes or Winnow are applied to the vectorized sequences. Li et al. [38] used the properties of singular value decomposition (SVD) of motion data to reduce multi-attribute motion data of different lengths and then obtain a feature vector for each motion matrix. By applying support vector machines (SVM) to the feature vectors, they efficiently classify and recognize real-world multi-attribute motion data. Lesh et al. [35] developed a new feature mining technique of sequences to reform the traditional methods, which ignore the potentially useful sequential patterns from large data. Lesh et al. [36] also proposed another feature mining technique called “scalable feature mining,” which is the improved FeatureMine algorithm for sequential data to act as the preprocessor to select features for standard classification algorithm such as Winnow and Naïve Bayes. Tseng and Lee [51] proposed a data mining method, named Classify-By-Sequence (CBS), to classify large time-series datasets. Legrand et al. [33] developed a classifier based on dynamic time warping (DTW). Their

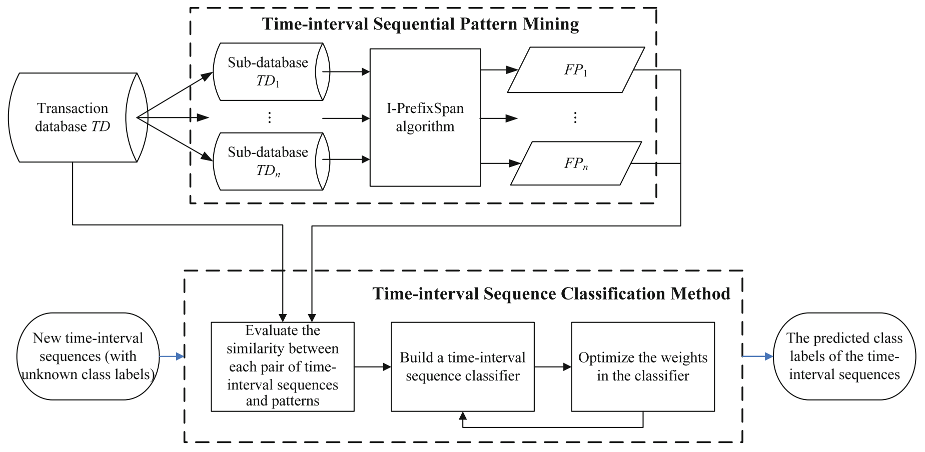
Time-interval Sequential Pattern Mining
Transaction database TD
Sub-database TD_1
Sub-database TD_n
I-PrefixSpan algorithm
FP_1
FP_n
Time-interval Sequence Classification Method
New time-interval sequences (with unknown class labels)
Evaluate the similarity between each pair of time-interval sequences and patterns
Build a time-interval sequence classifier
Optimize the weights in the classifier
The predicted class labels of the time-interval sequences

Fig. 1 Framework of the proposed time-interval sequence classification model

proposed classifier outperforms than the conventional methods such as Bayesian methods or neural networks in the experiment result. Exarchos et al. [19] used roll optimization to generate the optimal sets of weights through a progressive method. Nevertheless, the solution of their classification accuracy of the classification model is not efficient enough since the iterative number of roll optimization method is very large, which leads heavy computation of the objective function.

3 The time-interval sequence classification framework

The existing temporal data classification methods were mostly based on statistical methods [41], neural networks [34], or similarity-based techniques [58]. They focused on building classifiers with high accuracy; however, they were in lack of readable classification rules [20,52]. Therefore, to increase the readability of the classification results, a two-stage time-interval sequence classification framework, as shown in Fig. 1, is proposed. In the first stage, the customer transaction database *TD* is divided into n subdatabases $\{TD_1, TD_2, \ldots, TD_n\}$ according to the class label each sequence belongs to. The I-PrefixSpan algorithm is then applied to TD_c and generates a time-interval sequential pattern set FP_c for all $c = 1, 2, \ldots n$. In the second stage, the similarity measurement between two time-interval sequences is defined first. Then, a time-interval sequence classifier is built based on the time-interval sequential patterns in FP_c and sequences in TD_c for all $c = 1, 2, \ldots n$ by using the weighted-majority voting approach. The particle swarm optimization (PSO) algorithm is developed to iteratively adjust the weights in the classifier so that the accuracy of the classifier can be maximized.

3.1 Time-interval sequential patterns and mining

Let $I = \{i_1, i_2, \ldots, i_m\}$ be the set of all items. Accordingly, a sequence S_a is denoted as $((a_1, t_1), (a_2, t_2) \ldots, (a_n, t_n))$ where $a_j \in I$ is an item, and t_j represents the time a_j happens where $1 \leq j \leq n$, and $t_{j-1} < t_j$ for $2 \leq j \leq n$. In a sequence, if items occurred at the same time, they are presented according to the alphabet order. A transaction is represented by

$<sid, s>$, where *sid* is the identifier of this transaction and s is a sequence. A transaction database *TD* is formed by a set of transactions $< sid, s >$. Additionally, let Δt be time interval between two successive items in a sequence, and T_r be a given constant for $1 \leq r \leq X-1$. Then the time interval is divided into $X + 1$ sections, where

I_0 denotes the time interval Δt satisfying $\Delta t = 0$,
I_1 denotes the time interval Δt satisfying $0 < \Delta t \leq T_1$,
I_r denotes the time interval Δt satisfying $T_{r-1} < \Delta t \leq T_r$ for $1 < r < X-1$, and
I_X denotes the time interval Δt satisfying $T_{X-1} < \Delta t < \infty$.

Therefore, the set of time intervals can be represented as $TI = \{I_0, I_1, I_2, \ldots, I_X\}$. For instance, if $T_1 = 4$, $T_2 = 8$, and $T_3 = 12$, then $TI = \{I_{0,} I_1, I_2, I_{3,} I_4\}$ where $I_0 : \Delta t = 0$, $I_1 : 0 < \Delta t \leq 4, I_2 : 4 < \Delta t \leq 8, I_3 : 8 < \Delta t \leq 12, I_4 : 12 < \Delta t < \infty$. Based on the above symbols, a sequence $S_B = (b_1, \&_1, b_2, \&_2, \ldots, \&_{m-1}, b_m)$ is a time-interval sequence if $b_i \in I$ for $1 \leq i \leq m$ and $\&_i \in TI$ for $1 \leq i \leq m-1$.

For a sequence $S_a = ((a_1, t_1), (a_2, t_2), \ldots, (a_n, t_n))$ and a time-interval sequence $S_b = (b_1, \&_1, b_2, \&_2, \ldots, \&_{m-1}, b_m)$, S_b is said to be contained in S_a or S_b is a time-interval subsequence of S_a if integers $1 \leq j_1 < j_2 < \cdots < j_m \leq n$ exist such that (1) $b_1 = a_{j_1}, b_2 = a_{j_2}, \ldots, b_m = a_{j_m}$; (2)$t_{j_i} - t_{j_{i-1}}$ satisfies the condition of interval $\&_{i-1}$ for $2 \leq i \leq m$. The total number of items in a time-interval sequence is referred to as the *length* of the sequence. Therefore, a time-interval sequential pattern whose length is m is referred to as a m-time-interval sequential pattern. For a given time-interval sequence S_b, the support count of the sequence in database *TD* is denoted as *support_count*$(S_b) = |\{< sid, s > \mid < sid, s > \in TD \wedge S_b$ is contained in $s\}|$. A time-interval sequence S_b is called a *time-interval sequential pattern* or a *frequent time-interval sequence* if the percentage of transactions in *TD* containing S_b is greater than or equals to the user-specified minimum support (called *min_sup*). Given a sequence database *TD* and *min_sup*, time-interval sequential pattern mining is to determine all the time-interval sequential patterns in *TD* whose supports are more than or equal to *min_sup*.

In this study, the I-PrefixSpan algorithm proposed by Chen et al. [12] is used to generate the time-interval sequential patterns since its high computation efficiency. Three important concepts used in the I-PrefixSpan algorithm are introduced as follows. First, given a sequence $S_a = ((a_1, t_1), (a_2, t_2), \ldots, (a_n, t_n))$ and a time-interval sequence $S_b = (b_1, \&_1, b_2, \&_2, \ldots, \&_{m-1}, b_m)$ where $m \leq n$, S_b is a time-interval prefix of S_a if and only if (1) $b_i = a_i$ for $1 \leq i \leq m$; (2) $t_i - t_{i-1}$ satisfies the condition of $\&_{i-1}$ for $1 \leq i \leq m-1$. Second, assume that S_b is a time-interval subsequence of sequence S_a and that $i_1 < i_2 < \cdots < i_m$ are the indexes of the elements in S_a which match the elements of S_b. A subsequence $S'_a = ((a'_1, t'_1), (a'_2, t'_2), (a'_3, t'_3), \ldots, (a'_p, t'_p))$ of S_a, where $p = m + n - -i_m$, is called a projection of S_a with respect to S_b if and only if (1) S_b is a time-interval prefix of S'_a; (2) the last $n - i_m$ elements of S'_a are the same as the last $n - i_m$ elements of S_a. Third, let $S'_a = ((a'_1, t'_1), (a'_2, t'_2), (a'_3, t'_3), \ldots, (a'_j, t'_j))$ be the projection of sequence S_a with respect to time-interval sequence $S_b = (b_1, \&_1, b_2, \&_2, \ldots, \&_{k-1}, b_k)$. Then sequence $S_c = ((a'_{k+1}, t'_{k+1}), (a'_{k+2}, t'_{k+2}), \ldots, (a'_j, t'_j))$ is called the postfix of S_a with respect to prefix S_b.

The pseudo-code of the I-PrefixSpan algorithm is described in Fig. 2. The α-projected database, denoted as $TD|_\alpha$, is defined by the collection of postfixes for the sequences in database *TD* with respect to α. A table *TI_Table* is used to store this type of relation, where column corresponds to an item and a row corresponds to a time interval in *TI*. Each cell *TI_Table* (I_i, f) in the table records the number of transactions in $TD|_\alpha$ that contain item f and the time difference between this item and the last item of α lies within I_i. Processing

```
Subroutine I-PrefixSpan($\alpha$, $l$, $TD|_\alpha$)
01 Scan $TD|_\alpha$ one time.
02 if $l = 0$, then find all frequent items in $TD|_\alpha$.
03     for every frequent item $f$, Append $f$ to $\alpha$ as $\alpha$'.
04     Output all $\alpha$'.
05 if $l > 0$, then construct TI_Table by scanning all the transactions in $TD|_\alpha$.
06     for every frequent cell TI_Table($I_i$, $f$), Append ($I_i$, $f$) to $\alpha$ as $\alpha$'.
07     Output all $\alpha$'.
08 for each $\alpha$', construct $\alpha$'-projected database $TD|_{\alpha'}$,
       and call I-PrefixSpan ($\alpha$', $l$+1, $TD|_{\alpha'}$).
```

Fig. 2 The pseudo-code of the I-PrefixSpan algorithm

every transaction in $TD|_\alpha$ sequentially enables the table to be formed and the cells that are frequent to be identified. If the cell *TI_Table* (I_i, f) is a frequent cell, (I_i, f) can be appended to α to yield a time-interval sequential pattern α' and to construct the α'-projected database $TD|_\alpha$. Recursively discovering the time-interval sequential patterns in $TD|_{\alpha'}$ finally yields all the time-interval sequential patterns in *TD*.

3.2 Time-interval sequence similarity measurement

Different to traditional sequence comparison, the sequence discussed in this research contains time-interval information. Therefore, the similarity between two time-interval sequences is re-defined based on the following two considerations [49]. First, the more the number of mismatched items between two time-interval sequences is, the lower the similarity between the two sequences is. For example, the similarity between $A \rightarrow I_1 \rightarrow B \rightarrow I_4 \rightarrow C$ and $A \rightarrow I_1 \rightarrow B \rightarrow I_4 \rightarrow D$ should be higher than the similarity between $A \rightarrow I_1 \rightarrow B \rightarrow I_4 \rightarrow C$ and $A \rightarrow I_1 \rightarrow E \rightarrow I_3 \rightarrow G$, since the former case has only one mismatched item and later case has two mismatched items. Second, the similarity between time intervals is taken into account if and only if they have the same priori item. For example, there are two time-interval sequences $A \rightarrow I_1 \rightarrow B \rightarrow I_4 \rightarrow C$ and $A \rightarrow I_2 \rightarrow D \rightarrow I_3 \rightarrow E$. The similarity between I_1 and I_2 will be examined since they have the same priori item (i.e., item A). On the other hand, it is meaningless to evaluate the similarity between I_4 andI_3 since their priori items B and D are different.

Let $S_a = (a_1, \&_1^a, a_2, \&_2^a, \ldots, a_{v-1}, \&_{v-1}^a, a_v)$ and $S_b = (b_1, \&_1^b, b_2, \&_2^b, \ldots, b_{w-1}, \&_{w-1}^b, b_w)$ be the two time-interval sequences. Based on the above two considerations, the similarity measurement between S_a and S_b, denoted as $Sim_{a,b}$, is formulated as:

$$Sim_{a,b} = \sum_{p=1}^{L} (1 - Cost_{a,b,p})/L. \quad (1)$$

$L = \max(|S_a|, |S_b|)$ is the maximal length of S_a and S_b, $Cost_{a,b,p}$ is the cost that changes $\left(b_p, \&_p^b\right)$ in S_b to $\left(a_p, \&_p^a\right)$ in S_a, which can be derived using the proposed cost evaluation algorithm discussed later. In addition, the time-interval dissimilarity between $\&_p^b$ and $\&_p^a$ is defined as

$$\varphi_{a,b,p} = \frac{\left| f(\&_p^b) - f(\&_p^a) \right|}{f(I_X) - f(I_0) + 1} \tag{2}$$

where $f(\&_p^b)$ is the rank of the time-interval $\&_p^b$ in time-interval set *TI* and is defined as $f(I_r) = r$ where $r = 0, 1, \ldots, X$. Note that I_0 is inserted to the last position of the two time-interval sequences so that the last items in the two time-interval sequences can be compared with others.

The cost evaluation algorithm calculates the minimal total cost, also called the edit distance or Levenshtein distance, that transforms $S_b = (b_1, \&_1^b, b_2, \&_2^b, \ldots, b_{w-1}, \&_{w-1}^b, b_w)$ to $S_a = (a_1, \&_1^a, a_2, \&_2^a, \ldots, a_{v-1}, \&_{v-1}^a, a_v)$ using the dynamic programming approach. During the transformation, four basic edit operations of "no change," "substitution," "deletion," and "insertion" are used. The cost of taking required edit operations that change $(b_p, \&_p^b)$ in S_b to $(a_p, \&_p^a)$ in S_a is defined as:

$$Cost_{a,b,p} = \begin{cases} 1 & \text{if insertion or deletion} \\ 1 & \text{if substitution (with different priori items)} \\ 0 + \zeta \times \varphi_{a,b,p} & \text{if substitution (with the same priori item)} \\ 0 & \text{if no change} \end{cases} \tag{3}$$

where $\varphi_{a,b,p}$ is defined in Eq. (2) and ζ is the maximum degree of time-interval affection. In this study, ζ is recommended as the value less than 0.5. The pseudo-code of the proposed cost evaluation algorithm is shown in Fig. 3. The inputs to the algorithm are the time-interval sequences S_a and S_b, the maximum degree of time-interval affection ζ, and the time-interval set *TI*. The output is the cost vector recording the cost value in each position of Eq. (3). This algorithm is similar to the work of Tsai et al. [48] but different in *Time_Interval_Cost*( ) function in line 22.

For example, there are two time-interval sequences $S_a = A \rightarrow I_1 \rightarrow C \rightarrow I_2 \rightarrow D \rightarrow I_3 \rightarrow G \rightarrow I_1 \rightarrow E$ and $S_b = A \rightarrow I_1 \rightarrow C \rightarrow I_3 \rightarrow E \rightarrow I_2 \rightarrow G \rightarrow I_1 \rightarrow H \rightarrow I_4 \rightarrow J$ where $TI = \{I_0, I_1, I_2, I_3, I_4, I_5\}$. In addition, the maximum degree of time-interval affection ζ is set as 0.3. Note that I_0 is inserted after the last item in the two sequences before conducting similarity evaluation. Based on similarity measurement in Eq. (1) and the cost evaluation algorithm in Fig. 3, the similarity value between S_a and S_b is $((1-0)+(1-0.05)+(1-1)+(1-0)+(1-1)+(1-1))/6 = 0.49167$. Note that the cost of transforming C$\rightarrow$ I_3 in S_b to $C \rightarrow I_2$ in S_a is $Cost_{a,b,2} = 0{+}0.3{\times}(|3{-}2|/(5{-}0{+}1)) = 0.05$ because $f(I_3) = 3$, $f(I_2) = 2$, and $f(I_5) = 5$.

3.3 A time-interval sequence classifier

In this section, the time-interval sequence classifier is introduced in details. As mentioned before, the time-interval sequential patterns in FP_c are frequent patterns appeared in TD_c. Therefore, this research takes the time-interval sequential patterns in FP_c to represent the behaviors of all sequences in class c. If a new time-interval sequence s is similar to the time-interval sequential patterns in FP_c, the new time-interval sequence s has more chance belonging to class c. However, not all time-interval sequential patterns have the same classification judgment capability. Thus, each time-interval sequential pattern in FP_c should be assigned an important weight to reflect its own importance. Similarly, different frequent pattern set FP_c might have different affection to the class prediction result since the

```
Algorithm Cost_Evaluation($S_i$, $S_j$, $\zeta$, $TI$)

for (i = 1; i ++; i <= m)
    C[i][0] = i;
for (j = 1; j ++; j <= n)
    C[0][j] = j;
for (i = 1; i ++; i <= m){          //Starting the sequential pattern matching
    for (j = 1; j ++; j <= n){
        a1 = 0; a2 = 0; a3 = 0;
        if ($S_j$ [i-1] == $S_i$ [j-1]) {
            a3 = C[i-1][j-1];    // Edit operation: No Change
        } else {
            a1 = C[i-1][j]+1;    //Edit operation: Insertion ($\rightarrow$)
            a2 = C[i][j-1]+1;    //Edit operation: Deletion ($\downarrow$)
            a3 = C[i-1][j-1]+1; //Edit operation: Substitution ($\searrow$)
        }
        C[i][j] = min (a1,a2,a3); //Finish the matrix of edit distance
    }
}
q = max(m, n);                      //Derive longest sequence q
for (p = q-1; p --; p >= 0){        //Trace back for each position
   B[p] = Trace_Operation(C);
   if (B[p] == "Sub1")              // Edit operation: No Change
        Cost[p] = Time_Interval_Cost($S_i$ [p], $S_j$ [p] , $\zeta$, $TI$);
   else if(B[p] == "Sub2")          //Edit operation: Substitution ($\searrow$)
        Cost [p] = 1;
   else if (B[p] == "Del")          //Edit operation: Deletion ($\downarrow$)
        Cost [p] = 1;
   else (B[p] == "Ins")             //Edit operation: Insertion ($\rightarrow$)
        Cost [p] = 1;
}
return Cost;
```

Fig. 3 The pseudo-code of the proposed cost evaluation algorithm

representatives of sequential patterns in FP_c is different. Therefore, each frequent pattern set FP_c should be assigned an important weight to reflect its judgment importance also.

Based on the above concept, the following notations are defined. Let $fp_{c,k}$ be the kth time-interval sequential pattern in FP_c, $pw_{c,k}$ be the important weight of $fp_{c,k}$, and cw_c be the important weight of FP_c. According to Eq. (1), the similarity between time-interval sequential pattern $fp_{c,k}$ and a new time-interval sequence s, denoted as $Sim(fp_{c,k}, s)$, can be derived. Therefore, based on majority voting scheme, the time-interval sequence classifier is modeled as:

$$\text{Classifier}(x) = \underset{c=1,2,\ldots,n}{\arg\max}\left[cw_c \times \left(\sum_{k=1,2,\ldots,|FP_c|} \frac{pw_{c,k} \times Sim(fp_{c,k}, s)}{|FP_c|}\right)\right] \tag{4}$$

All weights are assigned as 1 at the initial stage to indicate they are equally important, but will be automatically adjusted by the particle swarm optimization (PSO) algorithm later.

A labeled instance is a pair $< s, y >$ where s is a time-interval sequence and y is the discrete class label associated with x where $y \in \{1, 2, \ldots, n\}$. Let a testing set *TS'* is a set of labeled instances where $TS' = \{< s_1, y_1 >, < s_2, y_2 > \cdots < s_m, y_m >\}$. The prediction accuracy of the classifier is calculated as:

$$\text{Accuracy} = \frac{\sum_{j=1,\ldots,|TS'|} I(\text{Classifier}(s_j) == y_j)}{|TS'|} \tag{5}$$

where $I(\cdot)$ is an indicator function that returns the value 1 if its argument is true and 0 otherwise.

3.4 Particle swarm optimization

It is clear that the two sets of weights ($pw_{c,k}$ and cw_c) in Eq. (4) might dramatically affect the accuracy of the proposed time-interval sequence classifier. Therefore, this research employs particle swarm optimization (PSO) algorithm [26] to find the best weights. Unlike other evolutionary algorithms such as genetic algorithm or bacterial chemotaxis optimization, the coding procedure for the PSO algorithm is relatively simple and easy. In addition, the computation complexity and running time of the PSO are much lower [2]. Peng et al. [44] presented a novel text classifier using positive and unlabeled examples and used the particle swarm optimization (PSO) algorithm to evolve the weights of all classifiers generated in the iteration steps. Lin et al. [39] proposed a particle swarm optimization–based back-propagation network approach (PSOBPN) to obtain the suitable parameter settings for BPN and to select the beneficial subset of features that result in a better classification accuracy rate. Wang and Yang [56] proposed a PSO-based Bayesian Networks Learning algorithm (PSOBNL) that introduced two major modifications on velocity and position updating rules. The experiments showed that this method outperformed other heuristic methods such as GA (genetic algorithm) and classical binary PSO.

Given a training set *TS*, the objective function of the PSO optimization method is to maximize the accuracy of the classifier by determining the values of $pw_{c,k}$ and cw_c. The maximal value of the objective function is 1 if all sequences in *TS* are correctly classified. Therefore, the accuracy of the classifier defined in Eq. (5) is used as the fitness function in the PSO algorithm. That is,

$$\max f(x) = \frac{\sum_{j=1,\ldots,|TS|} I(\text{Classifier}(s_j) == y_j)}{|TS|} \tag{6}$$

where x is one of the particles in the PSO algorithm that represents the values of two sets of weights ($pw_{c,k}$ and cw_c). Therefore, the ith particle is represented as

$$\begin{aligned} x_i = \big[&pw_{1,1}, pw_{1,2}, \ldots, pw_{1,|FP_1|}, pw_{2,1}, pw_{2,2}, \ldots, pw_{2,|FP_2|}, \ldots, \\ &pw_{n,1}, pw_{n,2}, \ldots, pw_{n,|FP_n|}, cw_1, cw_2, \ldots, cw_n\big] \end{aligned} \tag{7}$$

where the total dimension of the particle is $D = (\sum_{c=1}^{n} |FP_c|) + n$. To make the following explanation easier, the ith particle is represented as $x_i = [x_{id}]$ where $d = 1, 2, \ldots, D$. In the PSO algorithm, each particle moves depending on the best position the current particle found so far, denoted as *pbest* or $P_i = [p_{id}]$, and the global best position identified from the whole population (or within the neighborhood), denoted as *gbest* or $P_g = [p_{gd}]$. Shi and Eberhart [46] called the best previous experience (*pbest*) as the cognition part, and the best experience (*gbest*) as the social part. The ith particle changes its position by the given velocity $V_i = [v_{id}]$. The velocity of ith particle in position d is updated according to Eq. (8)

```
Algorithm PSO_method($TS$, $pn$, $v_{max}$, $w$, $c_1$, $c_2$)
01  Set $x_{i,d}^0$ as 1 for $i$ = 1 to $pn$, $d$ =1 to $D$;
02  Set $v_{i,d}^0$ as a random value in [-$v_{max}$, $v_{max}$] for $i$ = 1 to $pn$, $d$ =1 to $D$;
03  Calculate the fitness function $f(x_i^0)$ for $i$ = 1 to $pn$;
04  $t = 1$;
05  while (stopping criteria is not reached){
06      for ($i$ = 1; $i$++; $i$ <= $pn$) {
07          if ( $f(x_i^t)$ >= $f(x_i^{t-1})$) {
08              $p_{id}^t = x_{id}^t$ , for $d$ = 1, 2 … , $(\sum_{c=1}^{n} q_c) + n$ ;   // Set the local best value
09          }
10      }
11      for ($i$ = 1; $i$++; $i$ <= $pn$) {
12          if ( $f(x_i^t) == \max_{i=1,\ldots pn} f(x_i^t)$) {
13              $p_{gd}^t = x_{id}^t$ , for $d$ = 1, 2 … , $(\sum_{c=1}^{n} q_c) + n$ ;   // Set the global best value
14          }
15      }
16      for ($i$ = 1; $i$++; $i$ <= $pn$) {
17          $v_{id}^t = w v_{id}^{t-1} + c_1 r_1( p_{id}^t - x_{id}^t ) + c_2 r_2( p_{gd}^t - x_{id}^t )$;   // update $v_{id}^t$
18          $x_{id}^t = x_{id}^{t-1} + v_{id}^{t-1}$;   // update $x_{id}^t$
19      }
20      $t = t + 1$;
21  }
22  return the particle with highest fitness value;
```

Fig. 4 The pseudo-code of the PSO method for weight adjustment

every iteration, while the position d of the ith particle is updated as Eq. (9) in the search space.

$$v_{id}^t = wv_{id}^{t-1} + c_1 r_1 (p_{id}^t - x_{id}^t) + c_2 r_2 (p_{gd}^t - x_{id}^t) \quad \text{for} \quad d = 1, 2 \ldots D \tag{8}$$

$$x_{id}^{t+1} = x_{id}^t + v_{id}^t \quad \text{for} \quad d = 1, 2 \ldots D \tag{9}$$

where w is the inertia weight; c_1 indicates the cognition learning factor; c_2 indicates the social learning factor, and r_1 and r_2 are the random numbers that allow to maintain the diversity of the population and are uniformly distributed in U(0, 1). Inertia weight w is brought into PSO to balance the global and local search ability. A large w facilitates a global search and a small w eases a local search [17]. c_1 and c_2 are positive constants and usually are set as 2.

The pseudo-code of the proposed PSO method for weight adjustment is shown in Fig. 4. The input to the method includes the training set *TS* and the parameters for the PSO method of pn, v_{max}, w, c_1, and c_2 where pn is the particle population and v_{max} is the velocity restriction of particles. The output of the PSO method is the particle with highest fitness value so that the accuracy of the proposed classifier is highest. Variable t is the iteration number in the algorithm. As shown in lines 1 and 2, $x_{i,d}^0$ is set as 1 and v_{id}^0 is set as a random value ranging from -v_{max} to v_{max} for all i and d to keep off trapping in a local area. Subsequently, as shown from lines 4 to 21, x_i^t are iteratively modified until the stopping criterion is reached. The stopping criterion is reached when the error of all objective values obtained by the particles within one iteration is smaller than 5 % (i.e., $STD <= 0.05$).

Table 1 The dataset of the example I

sid	s_i	c_i
1	$\langle(A,3)(B,5)(C,25)(C,35)(D,72)\rangle$	1
2	$\langle(A,5)(B,7)(E,41)(C,70)\rangle$	1
3	$\langle(B,4)(C,6)(A,37)(A,65)\rangle$	1
4	$\langle(A,10)(C,49)(D,52)(C,106)\rangle$	1
5	$\langle(B,2)(A,27)(B,35)\rangle$	1
6	$\langle(A,1)(C,36)(D,61)(D,70)\rangle$	1
7	$\langle(A,2)(B,7)(C,20)(D,25)\rangle$	1
8	$\langle(A,3)(B,15)(D,17)\rangle$	1
9	$\langle(B,7)(D,21)(D,44)(B,53)\rangle$	2
10	$\langle(A,6)(A,13)(D,25)(D,46)\rangle$	2
11	$\langle(B,8)(B,32)(D,43)(C,79)(D,85)\rangle$	2
12	$\langle(B,1)(D,8)(A,22)(D,58)\rangle$	2
13	$\langle(C,3)(B,17)(D,29)(C,41)\rangle$	2
14	$\langle(C,12)(B,18)(D,23)\rangle$	2
15	$\langle(B,12)(A,21)(D,23)(B26)(D,30)\rangle$	2
16	$\langle(B,3)(B,7)(C,12)(D,18)\rangle$	2
17	$\langle(C,3)(B,15)(E,21)\rangle$	3
18	$\langle(C,9)(B,17)(A,30)(E,39)\rangle$	3
19	$\langle(A,3)(B,10)(C,23)(E,37)\rangle$	3
20	$\langle(C,16)(A,28)(E,31)(A,45)\rangle$	3
21	$\langle(B,4)(C,32)(B,38)(E,50)(A,82)\rangle$	3
22	$\langle(B,4)(C,11)(E,29)(A,63)\rangle$	3
23	$\langle(A,1)(C,5)(E,8)(A,16)\rangle$	3
24	$\langle(C,9)(B,12)(E,16)(A,32)\rangle$	3

4 Experimental illustration

4.1 Example I

To show the computation process of the proposed time-interval sequence classification framework, a dataset shown in Table 1 is used as an example. Each sequence in the dataset is represented by the set of items $I = \{A, B, C, D, E\}$ and corresponding timestamps. There are totally 24 sequences in the dataset where a sequence belongs to one of three class labels ($c = 1, 2, 3$). In each class, 6 sequences are randomly selected as testing data and others are training data. Therefore, the training dataset *TD* consist of 18 sequences and the testing dataset *TD'* consists of 6 sequences. In addition, according to the timestamps of sequences in the dataset, the set of time intervals is set as $TI = \{I_0, I_1, I_2, I_3, I_4, I_5\}$, where $I_0 : t = 0$, $I_1 : 0 < t \leq 10$, $I_2 : 10 < t \leq 20$, $I_3 : 20 < t \leq 30$, $I_4 : 30 < t \leq 40$, and $I_5 : 40 < t < \infty$.

In the first stage of the proposed time-interval sequence classification framework, the training dataset *TD* is divided into 3 subdatabases $\{TD_1, TD_2, TD_3\}$ according to the class label of each sequence. Then, the I-PrefixSpan algorithm is applied to TD_c and generates a time-interval sequential pattern set FP_c for all c. If *min_sup* is set as 2, the generated

Table 2 Time-interval sequential patterns in each class

FP_1		FP_2		FP_3	
$fp_{1,1}$	$A \to I_1 \to B$	$fp_{2,1}$	$A \to I_1 \to D$	$fp_{3,1}$	$A \to I_1 \to E$
$fp_{1,2}$	$A \to I_4 \to C$	$fp_{2,2}$	$A \to I_2 \to D$	$fp_{3,2}$	$B \to I_1 \to E$
$fp_{1,3}$	$A \to I_5 \to D$	$fp_{2,3}$	$A \to I_4 \to D$	$fp_{3,3}$	$B \to I_2 \to A$
$fp_{1,4}$	$B \to I_2 \to C$	$fp_{2,4}$	$B \to I_1 \to C$	$fp_{3,4}$	$B \to I_3 \to E$
$fp_{1,5}$	$C \to I_1 \to D$	$fp_{2,5}$	$B \to I_1 \to D$	$fp_{3,5}$	$B \to I_5 \to A$
$fp_{1,6}$	$C \to I_4 \to D$	$fp_{2,6}$	$B \to I_2 \to D$	$fp_{3,6}$	$C \to I_1 \to B$
$fp_{1,7}$	$A \to I_1 \to B \to I_2 \to C$	$fp_{2,7}$	$B \to I_4 \to D$	$fp_{3,7}$	$C \to I_2 \to E$
$fp_{1,8}$	$A \to I_4 \to C \to I_4 \to D$	$fp_{2,8}$	$B \to I_5 \to C$	$fp_{3,8}$	$C \to I_3 \to A$
		$fp_{2,9}$	$B \to I_5 \to D$	$fp_{3,9}$	$C \to I_5 \to A$
		$fp_{2,10}$	$C \to I_1 \to D$	$fp_{3,10}$	$E \to I_2 \to A$
		$fp_{2,11}$	$D \to I_1 \to B$	$fp_{3,11}$	$E \to I_4 \to A$
		$fp_{2,12}$	$D \to I_3 \to D$	$fp_{3,12}$	$C \to I_1 \to B \to I_2 \to A$
		$fp_{2,13}$	$A \to I_2 \to D \to I_3 \to D$	$fp_{3,13}$	$C \to I_2 \to E \to I_4 \to A$
		$fp_{2,14}$	$B \to I_1 \to C \to I_1 \to D$		
		$fp_{2,15}$	$B \to I_5 \to C \to I_1 \to D$		

Table 3 The similarities between sequential patterns in FP_1 and sequences in *TD*

	s_1	s_3	s_4	s_6	s_7	...	s_{21}	s_{22}	s_{24}
$fp_{1,1}$	0.38	0.72	0.21	0.21	0.48	...	0.38	0.50	0.24
$fp_{1,2}$	0.36	0.20	0.49	0.46	0.45	...	0.19	0.22	0.50
$fp_{1,3}$	0.36	0.72	0.44	0.48	0.45	...	0.60	0.50	0.50
$fp_{1,4}$	0.59	0.44	0.24	0.21	0.74	...	0.38	0.46	0.49
$fp_{1,5}$	0.80	0.46	0.70	0.71	0.48	...	0.40	0.49	0.25
$fp_{1,6}$	0.77	0.50	0.66	0.72	0.96	...	0.37	0.48	0.21
$fp_{1,7}$	0.59	0.44	0.46	0.21	0.74	...	0.40	0.46	0.24
$fp_{1,8}$	0.54	0.25	0.66	0.72	0.68	...	0.17	0.22	0.25

time-interval sequential pattern sets are shown in Table 2. In the second stage, the similarity between two time-interval sequences can be derived using Eq. (1) where the maximum degree of time-interval affection ζ in Eq. (3) is set as 0.3. Table 3 shows the similarities between sequential patterns in FP_1 and all time-interval sequences in *TD*.

To show the performance of the proposed time-interval sequence classifiers, the following two classifiers (i.e., classifier I and classifier II) are conducted. In classifier I, the PSO method is not applied so that the two set of weights ($pw_{c,k}$ and cw_c) in the classifier are all set as 1. In classifier II, the PSO method is used to automatically adjust the two set of weights. The set of parameters in the PSO method is set as follows: $pn = 20$; $v_{max} = 2.5$; $w = 0.5+$ Rand(0,1)/2; $c_1 = 1.5$; and $c_2 = 1.5$. After performing the PSO method, the final weight vector in the classifier II is [1.1941, 1.0806, 0.9217, 1.0010,..., 0.9677, 1.0438, 1.1897, 0.9814]. Note that the last three elements in the vector indicate that $cw_1 = 1.0438$, $cw_2 = 1.1897$, $cw_3 = 0.9814$.

Table 4 The classification results for the two classifiers

s_i	c_i	Predicted class using classifier I	Predicted class using classifier II
s_1	1	1	1
s_3	1	3	1
s_4	1	1	1
s_6	1	3	1
s_7	1	1	1
s_8	1	2	2
s_9	2	3	2
s_{10}	2	2	2
s_{11}	2	2	2
s_{14}	2	2	2
s_{15}	2	2	2
s_{16}	2	1	1
s_{17}	3	3	3
s_{18}	3	3	3
s_{20}	3	3	3
s_{21}	3	3	3
s_{22}	3	3	3
s_{24}	3	3	3

Table 5 Classification accuracy for the training dataset

Experiments	Prediction accuracy using classifier I (%)	Prediction accuracy using classifier II (%)	Improvement (%)
I	72.22	88.89	16.67
II	61.11	66.67	5.56
III	55.56	83.33	27.77
IV	61.11	77.78	16.67
Average	62.50	79.17	16.67
Max	72.22	88.89	27.77

Table 4 shows the classification results of the two classifiers. Based on the result, the prediction accuracy of the classifier I is $(3 + 4 + 6)/18 = 13/18 = 72.22\,\%$ and the predication accuracy of the classifier II is $(5 + 5 + 6)/18 = 16/18 = 88.89\,\%$ for the training dataset *TD*. The reasons is that sequences $s_3 =< (B, 4)(C, 6)(A, 37)(A, 65) >$, $s_6 =< (A, 1)(C, 36)(D, 61)(D, 70) >$, and $s_9 =< (B, 7)(D, 21)(D, 44)(B, 53) >$ incorrectly classified in classifier I are now correctly classified in classifier II. Thus, the classification accuracy for the training dataset increases 16.67 % after the PSO method is applied. However, when the testing dataset is evaluated, the classification accuracy for classifier I and classifier II is both 83.33 %.

Table 6 Classification accuracy for the testing dataset

Experiments	Prediction accuracy using classifier I (%)	Prediction accuracy using classifier II (%)	Improvement (%)
I	83.33	83.33	0.00
II	33.33	50.00	16.67
III	50.00	50.00	0.00
IV	66.67	83.33	16.66
Average	58.33	66.67	8.33
Max	83.33	83.33	16.67

Table 7 The basic information of the dataset in the example II

Class label	Industry type	Number of customers
1	Education	9
2	Finance and insurance	18
3	Traditional industry	18
4	Technology	15

To confirm the reliability of the proposed framework, k-fold cross-validation, where $k = 4$, is employed to the above process. Table 5 shows the classification accuracy for the two classifiers when training dataset is evaluated, while Table 6 shows the classification accuracy when testing dataset is evaluated. Based on the tables, the average improvement of the classification accuracy is 16.67 % for the training dataset and 8.33 % for the testing dataset. In addition, both tables indicate that the classifier II adopting the PSO method is significantly better than the classifier I without adopting the PSO method.

4.2 Example II

Another dataset contributed from the NorthWind database of Microsoft is used to demonstrate how the system parameters in the proposed time-interval sequence classification framework affect the classification result. This dataset consists of 341 order sequences generated by 60 customers within one year time period (from 4th July, 1996 to 3rd July, 1997). Since no class label information is provided for customers at the original database, this study takes the industry type of a customer as its class label. Table 7 shows the relationship between class label, industry type, and the number of customers in the dataset.

The system parameters that might affect the performance of the proposed system include two parameters in the I-PrefixSpan algorithm (i.e., the minimum support value *min_sup* and the set of the time interval *TI*) and four parameters in the PSO method (the population of particles *pn*, the cognitive and social learning factor c_1 and c_2, inertial constant w, and moving velocity v_{max}). Therefore, a set of experiments are designed to observe their affections. In the following discussion, the initial settings of system parameters are summarized in Table 8. Note that each experiment is evaluated using k-fold cross-validation where $k = 3$.

- The set of time intervals

The set of time intervals *TI* in the first stage of the proposed framework might affect the result of generated time-interval sequential patterns in the I-PrefixSpan algorithm. Therefore,

Table 8 The initial system parameter settings

Parameter	Initial value
pn	20
c_1	1.5
c_2	1.5
w	0.5+ rand( )/2
v_{max}	2.5
min_sup	70%
TI	$TI = \{I_0, I_1, I_2, I_3, I_4,\}$ where $I_0 : t = 0, I_1 : 0 < t \leq 90, I_2 : 90 < t \leq 180, I_3 : 180 < t \leq 270, I_4 : 270 < t \leq \infty$

Table 9 Classification accuracy with different *TI*

Experiment	$TI = \{I_0, I_1, I_2, I_3\}$		$TI = \{I_0, I_1, I_2, I_3, I_4\}$		$TI = \{I_0, I_1, I_2, I_3, I_4, I_5, I_6\}$	
	Without optimization (%)	With optimization (%)	Without optimization (%)	With optimization (%)	Without optimization (%)	With optimization (%)
I	32.50	40.00	35.00	42.50	20.00	30.00
II	25.00	42.50	22.50	45.00	17.50	42.50
III	17.50	35.00	20.00	42.50	17.50	35.00
Average	25.00	39.17	25.83	43.33	18.33	35.83

the following three time-interval settings are tested, while other parameters remain the same as the ones in Table 8:

1. $TI = \{I_0, I_1, I_2, I_3\}$, where $I_0 : t = 0, I_1 : 0 < t \leq 120, I_2 : 120 < t \leq 240, I_3 : 240 < t < \infty$.
2. $TI = \{I_0, I_1, I_2, I_3, I_4\}$, where $I_0 : t = 0, I_1 : 0 < t \leq 90, I_2 : 90 < t \leq 180, I_3 : 180 < t \leq 270, I_4 : 270 < t < \infty$.
3. $TI = \{I_0, I_1, I_2, I_3, I_4, I_5, I_6\}$, where $I_0 : t = 0, I_1 : 0 < t \leq 60, I_2 : 60 < t \leq 120, I_3 : 120 < t \leq 180, I_4 : 180 < t \leq 240, I_5 : 240 < t \leq 300, I_6 : 300 < t < \infty$.

Table 9 exhibits the classification accuracy for the three time-interval settings. It is obvious the highest average classification accuracy is obtained when $TI = \{I_0, I_1, I_2, I_3, I_4\}$ and the PSO method is adopted. It is interesting that the classification accuracy when $TI = \{I_0, I_1, I_2, I_3, I_4, I_5, I_6\}$ is worst. The reason is that, if the time intervals are too many, generated sequential patterns are not strong enough to represent the behaviors in a specific class. Therefore, the set of time intervals is suggested as $\{I_0, I_1, I_2, I_3, I_4\}$.

- The minimum support value

The *min_sup* is a critical value affecting the number of time-interval sequential patterns generated by I-PrefixSpan algorithm. Table 10 summarizes the classification accuracy when *min_sup* is set as 50, 60 and 70 % (i.e., 30, 36, and 42), while other parameters remain the same as the ones in Table 8. It is obvious, when $min\_sup = 70\,\%$, the average classification accuracy with optimization process is the highest. Originally, we expect that a lower

Table 10 Classification accuracy with different *min_sup* values

Experiment	*min_sup* = 50%		*min_sup* = 60%		*min_sup* = 70%	
	Without optimization (%)	With optimization (%)	Without optimization (%)	With optimization (%)	Without optimization (%)	With optimization (%)
I	27.50	32.50	27.50	32.50	35.00	42.50
II	25.00	37.50	22.50	37.50	22.50	45.00
III	22.50	40.00	20.00	40.00	20.00	42.50
Average	25.00	36.67	23.33	36.67	25.83	43.33

min_sup value makes the classification accuracy higher, since more sequential patterns are generated by the I-PrefixSpan algorithm. However, this experiment result shows that using more sequential patterns to represent sequences in classes does not guarantee higher classification accuracy. Based on the above observation, the minimum support value is recommended as 70 % for this dataset.

- The population of particles

It is suggested that a good solution might be obtained if the population of particles *pn* in the PSO algorithm is set as 20 or 30 [26]. However, since the dimension of a particle in this study is relatively large, the population of particles is tested using 20, 50, and 80, while other parameters remain the same as the ones in Table 8. Figure 5 shows the classification accuracy without running the PSO algorithm and the ones after running the PSO algorithm with different *pn* values. It is clear that, after running the PSO algorithm, the classification accuracy is greatly improved. The highest average classification accuracy happened when $pn = 50$. Note that, when *pn* is 80, the classifier is over-trained so that its average classification accuracy drops to 39.17 %. Based on the above observation, the population of particles is recommended as 50.

- The cognitive and social learning factors

In the PSO algorithm, the cognitive learning factor c_1 controls the acceleration of local search, while the social learning factor c_2 controls the acceleration of global search. Typically, c_1 and c_2 are recommended as 2 in many studies [17. 26]. In this experiment, five combinations as shown in Fig. 6 are conducted. It is clear the classification accuracy is highest when $c_1 = c_2 = 1.5$, $c_1 = c_2 = 2$, and $c_1 = 1$, $c_2 = 3$. However, the variance of the classification accuracy is larger when $c_1 = c_2 = 2$. Therefore, the cognitive learning factor c_1 and social learning factor c_2 are both set as 1.5.

- The inertia weight

The inertia weight w in the PSO algorithm balances the global exploration and local exploitation in the PSO algorithm. Global search performs well if a large inertia weight is applied, while a small inertia weight facilitates the local search performance. According to the literatures [23], inertia weight is recommended as the value ranging from 0.5 to 1. Therefore, in this experiment, inertia weight is set as 0.5, 0.5 + rand( )/2, and 1 where *rand*( ) generates a random number uniformly distributed within [0,1]. As shown in Fig. 7, the highest average classification accuracy appears when $w = 0.5+\text{rand( )}/2$. Therefore, inertia weight is suggested as 0.5+rand( )/2 in this case.

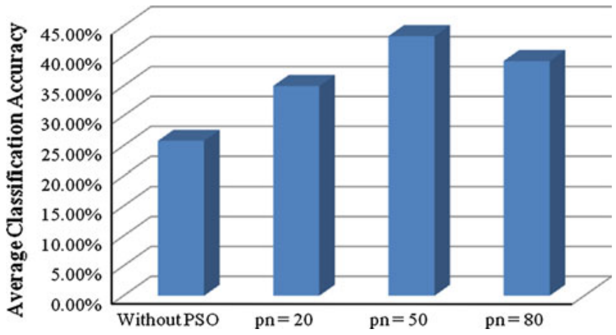

Fig. 5 Classification accuracy under different *pn* values

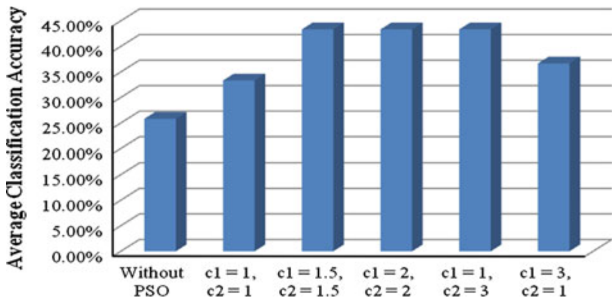

Fig. 6 Classification accuracy under different c_1 and c_2 values

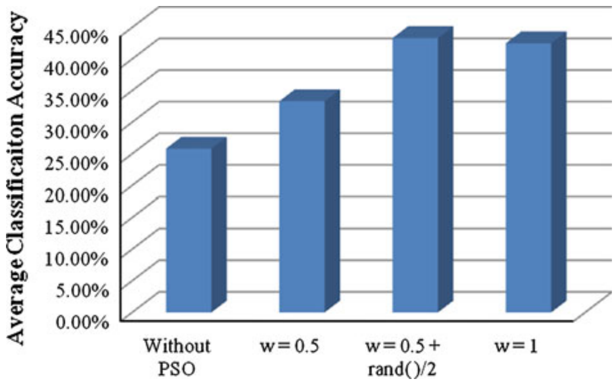

Fig. 7 Classification accuracy under different w values

- The maximum velocity

In the PSO algorithm, slower moving velocity of particles drives a better solution. However, when the moving velocity is too small, it is difficult for particles to reach global optimal solution. Oppositely, if moving velocity is too large, particles will move too fast and hard to reach convergence. Therefore, the moving velocity of a particle is restricted to the range of

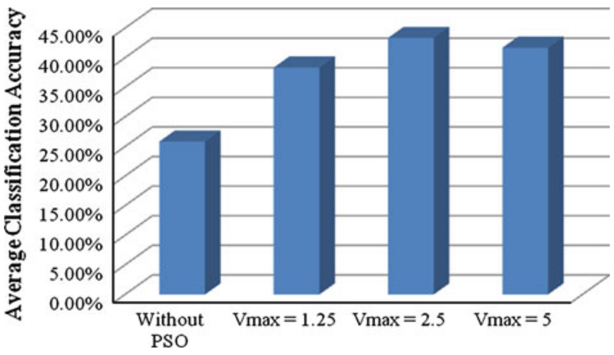

Fig. 8 Classification accuracy under different v_{max} values

Table 11 The computational time of the proposed classification framework (unit: seconds)

Experiments	First stage	Second stage
I	10.2078	166.5802
II	9.3813	187.8401
III	4.1797	289.7151
Average	7.9229	214.7118

The time in each experiment is the average value after 10 runs

$[-v_{max}, v_{max}]$. Based on the suggestion of Eberhart and Shi [17], this experiment tests v_{max} is used 1.25, 2.5, and 5. As shown in Fig. 8, the highest average classification accuracy is found when v_{max} is 2.5. Therefore, maximum velocity is suggested as 2.5 in this case.

Based on the above experiments, the system parameters are suggested as $min\_sup = 70\,\%$, $TI = \{I_0, I_1, I_2, I_3, I_4\}$, $pn = 50$, $c_1 = c_2 = 1.5$, $w = 0.5+$ rand( )/2, and $v_{max} = 2.5$. Based on these parameters, the average classification accuracy of the proposed framework is 43.33 %, while it is 25.83 % if no optimal process is performed. Although a great improvement in the average classification accuracy can be found for the proposed framework, the classification accuracy is not high. The major reason is that this study subjectively adopts the industry type of a customer as its class label, since no class information is provided at the original database. If a more representative attribute can be found and used as class label, the classification accuracy should increase dramatically.

Table 11 shows the computational time for the two stages of the proposed classification framework in the three experiments. The average time of the first stage in the proposed framework takes about 8 s to complete, while the average time of the second stage takes about 215 s to complete. It is obvious that the time of weight optimization in the second stage is significantly longer than the time of sequential pattern mining in the first stage. Figure 9 displays the more detail of the computational time of the 10 runs in the second stage. Among the 10 runs, the minimum computational time for experiments I–III is 50.89, 42.77, and 11.81 s respectively, while the maximum computational time is 357.33, 641.97, 768.08 s, respectively. The variance of the computational time in the second stage is large since the random characteristics of the PSO algorithm exist.

4.3 Validation

As the best we know after a complete literature survey, none of researches takes time-interval information between activities into consideration when conducting sequence classification.

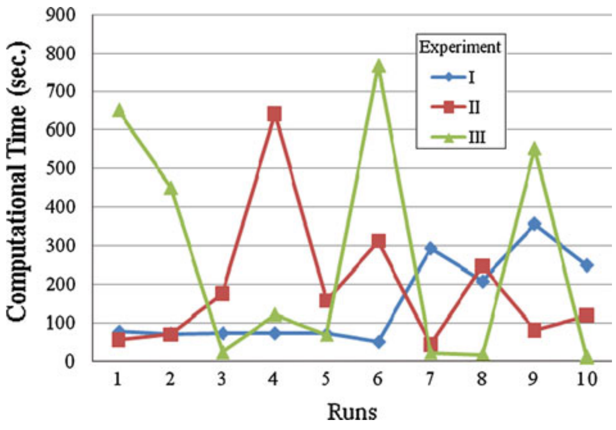

Fig. 9 The computational time for different experiments in the second stage

Therefore, validation for the proposed time-interval sequence classification method will be separated into two folds. The first fold is to compare the proposed sequence classification method optimized with PSO to the sequence classification method proposed by Exarchos et al. [19]. No time-interval information is considered in this fold for both methods. The work of Exarchos et al. was originally applied to evaluate the biological sequence dataset. The weight values in their sequence classification method are optimized by Roll optimization. Roll optimization method proposed by Evangelakis et al. [18] belongs to the class of pattern search methods. It resembles the obvious (and ad hoc) alternating variables method [21] and proceeds by exploring the local topology of the objective function and taking proper steps along each direction separately. Since no time-interval consideration is taken in Exarchos et al., the validation in the first fold will focus on classification accuracy comparison.

To fulfill the goal of classification accuracy comparison, a group of primary protein sequences derived from the Protein Data Bank (PDB) [5] is utilized. All data in this group correspond to a specific fold of the structural classification of proteins (SCOP) database [40]. In this validation, 1,000 proteins (sequences) belonging to 17 SCOP classes are retrieved. Two-third of them randomly selected from each class is formed as a training dataset, while the rest are formed as a testing dataset. In addition, six approaches are applied to evaluate the classification accuracy [19]:

- Approach 1 (App. 1): Set both pattern and class weights equal to 1 and calculate sequence similarity without optimization.
- Approach 2 (App. 2): Set the pattern weights equal to 1 ($pw = 1$) and maximize the prediction accuracy to find the optimal class weights cw^*.
- Approach 3 (App. 3): Set the class weights equal to 1 ($cw = 1$) and maximize the prediction accuracy to find the optimal pattern weights pw^*.
- Approach 4 (App. 4): From App. 3, set the pattern weights equal to the optimal ones pw^* and maximize the prediction accuracy to identify the optimal class weights cw^*.
- Approach 5 (App. 5): From App. 2, set the class weights equal to the optimal ones cw^* and maximize the prediction accuracy to identify the optimal pattern weights pw^*.
- Approach 6 (App. 6): Together optimize both pw and cw and find the optimal pattern weights pw^* and the optimal class weights cw^*.

Table 12 shows the classification accuracy of the two methods for the training dataset and testing dataset when $min\_sup = 0.5$. For the training dataset, the average classification

Table 12 Classification accuracy comparison using a protein sequence dataset

Approach	Training dataset		Testing dataset	
	Exarchos et al. method (%)	Proposed method (%)	Exarchos et al. method (%)	Proposed method (%)
App. 1	36.50	35.89	22.50	22.46
App. 2	58.70	56.76	31.40	34.13
App. 3	52.30	53.15	26.70	31.44
App. 4	54.40	57.36	27.50	33.23
App. 5	63.80	58.56	35.60	33.83
App. 6	57.50	52.85	31.70	35.63
Average	53.87	52.43	29.23	31.79

accuracy of Exarchos et al. method is 53.87 %, while the average classification accuracy of the proposed method is 52.43 %. The classification accuracy of the proposed method is 1.44 % lower than the one of Exarchos et al. method for the training dataset. However, for the testing data, the average classification accuracy of Exarchos et al. method is 29.23 %, while the average classification accuracy of the proposed method is 31.79 %. The classification accuracy of the proposed method is 2.56 % higher than the one of Exarchos et al. method for the testing dataset in average. Moreover, for App. 6 in which optimization is simultaneously conducted for the two sets of weights, the classification accuracy of the proposed method is 3.93 % (35.63–31.70 %) higher than the one of Exarchos et al. method. It indicates that the proposed method has better class prediction capability for unseen sequences. One possible reason is that the Exarchos et al. method optimized using Roll optimization tends to over-fit the sequence classification model while the proposed method optimized using PSO can resolve the over-fitting problem.

The second fold of the validation is to show the strengths and benefits of the proposed time-interval sequence classification method. Therefore, the proposed time-interval sequence classification method (called Model 2) is compared with the sequence classification method without time-interval consideration (called Model 1). In Model 1, a sequence is collected in the form of $S_a = (a_1, a_2, , \ldots, a_n)$ where a_n is an activity and a_{i-1} happens before a_i. In Mode 2, a sequence is collected in the form of $S_b = ((b_1, t_1), (b_2, t_2), \ldots, (b_n, t_n))$ where b_i is an activity and t_i is the timestamp b_i happens. Then, sequence S_b is changed as the time-interval sequence format of $S_b' = (b_1, \&_1, b_2, \&_2, \ldots, \&_{m-1}, b_m)$ where $\&_i \in \{I_0, I_1, I_2, \ldots, I_X\}$.

Three different size datasets are experimented to show the benefits of the proposed classification method. For small size dataset, the dataset in Sect. 4.1, which contains 24 sequences with 3 class labels, is tested. Table 13 shows the classification accuracy of Model 1 and Model 2 for small size dataset when PSO optimization is not applied for. The average classification accuracy for Model 1 is 50.00 % and the one for Model 2 is 58.33 %. Table 14 shows the classification accuracy of Model 1 and Model 2 for the same dataset when PSO optimization is applied for. The average classification accuracy for Model 1 is 62.50 % while the average classification accuracy for Model 2 is 66.67 %. Based on the two tables, the classification accuracy rates improve 8.33 and 4.17 % if time-interval information is considered in the sequence classification framework.

For medium size dataset, there are totally 200 sequences in the dataset where a sequence belongs to one of 4 class labels ($c = 1, 2, 3, 4$). Each sequence is randomly generated by the set of 11 activity items $I = \{A, B, C, \ldots, K\}$ with the length of [5,8]. The corresponding timestamp for each activity is ranging from [0, 100]. After transformation, time-intervals

Table 13 Classification accuracy for the small size dataset when PSO optimization is not applied for

Experiments	Sequence classification model without time-interval consideration (Model 1) (%)	The proposed time-interval sequence classification model (Model 2) (%)
I	66.67	83.33
II	50.00	33.33
III	33.33	50.00
IV	50.00	66.67
Average	50.00	58.33
Max	66.67	83.33

Table 14 Classification accuracy for the small size dataset when PSO optimization is applied for

Experiments	Model 1 (%)	Model 2 (%)
I	66.67	83.33
II	66.67	50.00
III	66.67	50.00
IV	50.00	83.33
Average	62.50	66.67
Max	66.67	83.33

between activities is transferred as $TI = \{I_0, I_1, I_2, I_3, I_4, I_5\}$, where $I_0 : t = 0$, $I_1 : 0 < t \leq 10$, $I_2 : 10 < t \leq 20$, $I_3 : 20 < t \leq 30$, $I_4 : 30 < t \leq 40$, and I_5: $40 < t < \infty$. Table 15 shows the average numbers of sequential patterns generated for each class label in the first stage of the classification framework. If time interval is not taken into consideration, PrefixSpan algorithm [43] is applied for generating the sequential patterns. On the other hand, if time-interval consideration is taken, I-PrefixSpan algorithm [12] is applied for generating time-interval sequential patterns. For both algorithms, when *min_sup* value increases, the average number of generated patterns decreases. In addition, the average number of patterns generated by I-PrefixSpan algorithm is larger than PrefixSpan algorithm except when *min_sup* is 35 %.

In the second stage, for each class label, 40 sequences are randomly selected as training data and 10 as testing data. Tables 16 and 17 show the classification accuracy of training data and testing data respectively for the medium size dataset under different *min_sup* values. It is observed that classification accuracy gradually decreases when *min_sup* values increases although it is not very significant. The lowest accuracy happens when the classification model without applying time interval and PSO optimization, while the highest accuracy happens when the classification model applying time interval and PSO optimization. Figure 10 shows the computational time for building Model 1 and Model 2 under different *min_sup* values. It takes 10.53 and 17.71 s for building Model 1 and Model 2 respectively when *min_sup* is 20 %. However, when *min_sup* is 35 %, the computational time for Model 2 is less than the one for Model 1 since the number of patterns in Model 2 is less than the one in Model 1.

For large size dataset, there are totally 500 sequences in the dataset where a sequence belongs to one of 5 class labels ($c = 1, 2, 3, 4, 5$). Each sequence is randomly generated by the set of 12 activity items $I = \{A, B, C, \ldots, L\}$ with the length of [8,12]. The corresponding timestamp for each activity is ranging from [0, 200]. After transformation, time intervals

Table 15 The average numbers of sequential patterns generated in the first stage for the medium size dataset

	min_sup (%)	Class 1	Class 2	Class 3	Class 4	Total
Without time-interval consideration (PrefixSpan algorithm)						
	20	48.50	57.50	57.50	46.75	210.25
	25	30.50	33.00	33.75	29.75	127.00
	30	24.00	24.50	26.00	23.50	98.00
	35	21.75	19.75	20.75	18.50	80.75
With time-interval consideration (I-PrefixSpan algorithm)						
	20	84.50	56.25	60.50	46.50	247.75
	25	57.50	34.25	32.75	25.25	149.75
	30	43.25	20.00	15.50	15.25	94.00
	35	33.50	12.00	6.25	9.50	61.25

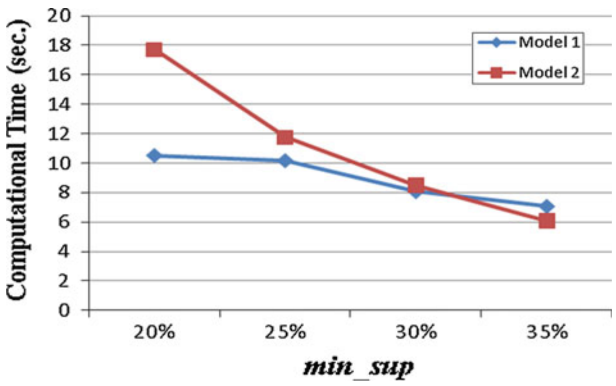

Fig. 10 The computational time for building Model 1 and Model 2 for the medium size dataset

Table 16 Classification accuracy of the training data for the medium size dataset

min_sup (%)	Model 1 without optimization (%)	Model 2 without optimization (%)	Model 1 with optimization (%)	Model 2 with optimization (%)
20	91.25	97.34	97.50	99.06
25	92.50	95.63	97.60	98.75
30	91.09	95.16	97.66	98.28
35	90.31	91.88	97.50	97.81

between activities is transferred as $TI = \{I_0, I_1, I_2, I_3, I_4, I_5\}$, where $I_0 : t = 0$, $I_1 : 0 < t \leq 20$, $I_2 : 20 < t \leq 40$, $I_3 : 40 < t \leq 60$, $I_4 : 60 < t \leq 80$, and $I_5 : 80 < t < \infty$. Table 18 shows the average numbers of sequential patterns generated for each class label in the first stage of the classification framework. Similarly, when *min_sup* value increases, the average number of generated patterns decreases for both algorithms. However, the average

Table 17 Classification accuracy of the testing data for the medium size dataset

min_sup (%)	Model 1 without optimization (%)	Model 2 without optimization (%)	Model 1 with optimization (%)	Model 2 with optimization (%)
20	91.25	97.50	92.50	98.13
25	90.00	95.00	88.75	96.88
30	93.75	94.38	91.88	96.88
35	92.50	91.88	91.25	92.50

Table 18 The average numbers of sequential patterns generated in the first stage for the large size dataset

	min_sup (%)	Class 1	Class 2	Class 3	Class 4	Class 5	Total
Without time-interval consideration (PrefixSpan algorithm)							
	20	104.50	124.5	112.00	129.75	100.00	570.75
	25	72.25	84.00	74.00	84.50	68.50	383.25
	30	40.25	48.25	47.25	51.75	40.25	227.75
	35	21.75	30.00	29.00	31.25	23.50	135.50
	40	16.75	19.00	17.00	20.25	15.00	88.00
With time-interval consideration (I-PrefixSpan algorithm)							
	20	118.00	156.5	83.75	77.25	75.00	510.50
	25	85.25	101.00	58.25	61.00	53.25	358.75
	30	56.75	61.50	38.25	43.75	31.00	231.25
	35	39.50	41.75	23.00	26.25	22.75	153.25
	40	32.50	32.00	18.25	16.25	18.25	117.25

number of patterns generated by I-PrefixSpan algorithm is less than the one by PrefixSpan algorithm when *min_sup* is less than 30 %.

In the second stage, for each class label, 75 sequences are randomly selected as training data and 25 as testing data. Table 19 shows the classification accuracy of testing data for the large size dataset under different *min_sup* values. The lowest accuracy happens when the classification model without applying time interval and PSO optimization, while the highest accuracy happens when the classification model applying time interval and PSO optimization. Figure 11 shows the computational time for building Model 1 and Model 2 under different *min_sup* values. It takes 42.42 s for building Model 1 and 37.16 s for Model 2 respectively when *min_sup* is 20 %. However, when *min_sup* is 35 and 40 %, the computational time for Model 2 is larger than the one for Model 1 since the number of patterns in Model 2 is larger than the one in Model 1.

Based on the experiment results of three different size datasets, it is clear that the proposed time-interval sequence classification method is superior to the method without time-interval consideration. In addition, when PSO optimization is applied, the classification accuracies of the three experiments improve significantly.

Table 19 Classification accuracy of the testing data for the large size dataset

min_sup	Model 1 without optimization	Model 2 without optimization	Model 1 with optimization	Model 2 with optimization
20	91.80	93.60	95.00	96.60
25	91.80	92.80	95.00	96.20
30	90.60	92.40	95.20	96.40
35	87.60	91.80	94.00	95.60
40	86.20	90.40	93.40	94.60

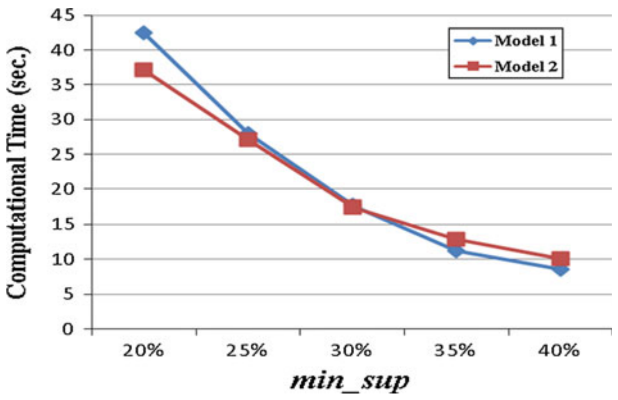

Fig. 11 The computational time for building Model 1 and Model 2 for the large size dataset

5 Conclusions

The importance of behavioral study from the informatics perspective is receiving increasing recognition. Behavior informatics, also known as behavior computing, aims at computing methodologies and techniques for handling behavior modeling, representation, dynamics, network analysis, impact analysis, and pattern analysis. When applying to customer relationship management (CRM), classification, one of the most popular behavior prediction tools in behavior informatics, can be used to build a customer behavior model and predict their future behaviors through classifying database records into a number of predefined classes. Although this behavior analysis tool can help business intelligence accumulate and enterprise operation boost, two major problems are identified. First, most previous CRM classification methods took demographic, RFM-type, or activity attributes as classification criteria and seldom took temporal relationship among these attributes into account. Second, the time-interval information between activities was not discussed in their classification frameworks so that customer behaviors cannot be correctly classified.

Therefore, the contribution of this research in behavior informatics is to take customer temporal behavior data, called time-interval sequences, as classification criteria and develop a two-stage customer temporal behavior classification framework. In the first stage, time-interval sequential patterns are discovered from customer temporal databases. The purpose of the first stage is to obtain less but representative time-interval sequences (sequential patterns) so that the computational speed of building a classifier in the second stage will be boosted. In the second stage, a new similarity measure and evaluation approach for time-interval sequences was developed. Then, a time-interval sequence classifier optimized by the particle swam optimization (PSO) algorithm is proposed to achieve high classification accuracy. With the proposed classification framework, a new customer can be easily and accurately predicted as one of previous known class based on his/her temporal behavior data. The comparison experiments show that classification accuracy of the proposed sequence classifier with time-interval consideration is much better than that without time-interval consideration.

Although the proposed time-interval sequence classification framework has been successfully implemented and tested well in this research, some works can be improved further. First, different time-interval sequence similarity measurements can be tried. For example, the editing distance between two sequences is computed through the dynamic programming algorithm in this study. It is suggested that variant string matching algorithms or strategies can be studied so that their influences can be revealed. Second, different optimization methods such as neural networks and genetic algorithms can be employed to obtain the optimized weights in the classifier. Finally, the developed time-interval sequence classification framework can be applied not only in CRM applications but also in stock market prediction, medical disease prediction, and anomaly manufacturing process detection.

Acknowledgments This work was partially supported by the National Science Council of Taiwan No. 100-2628-E-155-005-MY2.

References

1. Ahn H, Kim KJ, Han I (2006) Hybrid genetic algorithms and case-based reasoning systems for customer classification. Expert Syst 23(3):127–144
2. Arumugam MS, Rao MVC, Chandramohan A (2008) A new and improved version of particle swarm optimization algorithm with global–local best parameters. Knowl Inf Syst 16(3):331–357
3. Baesens B, Verstraeten G, Vanden Poel D, Egmont-Petersen M, Van Kenhove P, Vanthienen J (2004) Bayesian network classifiers for identifying the slope of the customer lifecycle of long-life customers. Eur J Oper Res 156(2):508–523
4. Baesens B, Viaene S, Van Den Poel D, Vanthienen J, Dedene G (2002) Bayesian neural network learning for repeat purchase modelling in direct marketing. Eur J Oper Res 138(1):191–211
5. Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IN, Bourne PE (2000) The protein data bank. Nucleic Acids Res 28(1):235–242
6. Bruyn AD, Liechty JC, Huizingh EKRE, Lilien GL (2008) Offering online recommendations with minimum customer input through conjoint-based decision aids. Mark Sci 27(3):1–38
7. Bouchaffra D, Tan J (2006) Structural hidden Markov models using a relation of equivalence: application to automotive designs. Data Min Knowl Discov 12(1):79–96
8. Cao L (2008) Behavior informatics and analytics: let behavior talk. In: Proceedings of IEEE international conference on data mining workshops, pp 87–96
9. Cao L (2010) In-depth behavior understanding and use: the behavior informatics approach. Inf Sci 180(17):3067–3085
10. Cao L, Yu PS (2009) Behavior informatics: an informatics perspective for behavior studies. IEEE Intell Inform Bull 10(1):6–11
11. Chen H, Chung W, Xu J, Wang G, Qin Y, Chau M (2004) Crime data mining: a general framework and some examples. Computer 37(4):50–56
12. Chen YL, Chiang MC, Ko MT (2003) Discovering time-interval sequential patterns in sequence databases. Expert Syst Appl 25(3):343–354
13. Cheung KW, Kwok JT, Law MH, Tsui KC (2003) Mining customer product ratings for personalized marketing. Decis Support Syst 35(2):231–243
14. Chiu C (2002) A case-based customer classification approach for direct marketing. Expert Syst Appl 22(2):163–168
15. Chu BH, Tsai MS, Ho CS (2007) Toward a hybrid data mining model for customer retention. Knowl Based Syst 20(8):703–718
16. Cui D, Curry D (2005) Prediction in marketing using the support vector machine. Mark Sci 24(4): 595–615

17. Eberhart RC, Shi Y (2001) Particle swarm optimization: developments, applications and resources. In: Proceedings of the IEEE conference on evolutionary computation, ICEC 1, pp 81–86
18. Evangelakis GA, Rizos JP, Lagaris IE, Demetropoulos IN (1987) Merlin—a portable system for multidimensional minimization. Comput Phys Commun 46(3):401–415
19. Exarchos TP, Tsipouras MG, Papaloukas C, Fotiadis DI (2008) A two-stage methodology for sequence classification based on sequential pattern mining and optimization. Data Knowl Eng 66(3):467–487
20. Exarchos TP, Tsipouras MG, Papaloukas C, Fotiadis DI (2009) An optimized sequential pattern matching methodology for sequence classification. Knowl Inf Syst 19(2):249–264
21. Fletcher R (1987) Practical methods of optimization. Wiley, New York
22. Ha SH, Bae SM, Park SC (2002) Customer's time-variant purchase behavior and corresponding marketing strategies: an online retailer's case. Comput Ind Eng 43(4):801–820
23. Huang CL, Dun JF (2008) A distributed PSO-SVM hybrid system with feature selection and parameter optimization. Appl Soft Comput 8(4):1381–1391
24. Jiang T, Tuzhilin A (2006) Segmenting customers from population to individuals: does 1-to-1 keep your customers forever. IEEE Trans Knowl Data Eng 18(10):1297–1311
25. Joung JG, June OS, Zhang BT (2006) Protein sequence-based risk classification for human papillomaviruses. Comput Biol Med 36(6):656–667
26. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of the IEEE international conference on neural networks, pp 1942–1948
27. Kim J, Suh E, Hwang H (2003) A model for evaluating the effectiveness of crm using the balanced scorecard. J Interact Mark 17(2):5–19
28. Kim SY, Jung TS, Suh EH, Hwang HS (2006) Customer segmentation and strategy development based on customer lifetime value: a case study. Expert Syst Appl 31(1):101–107
29. Kim Y, Street WN (2004) An intelligent system for customer targeting: a data mining approach. Decis Support Syst 37(2):215–228
30. Kim YH, Moon BR (2006) Multicampaign assignment problem. IEEE Trans Knowl Data Eng 18(3): 405–414
31. Köknar-Tezel S, Latecki LJ (2011) Improving SVM classification on imbalanced time series data sets with ghost points. Knowl Inf Syst 28(1):1–23
32. Lee TS, Chiu CC, Chou YC, Lu CJ (2006) Mining the customer credit using classification and regression tree and multivariate adaptive regression splines. Comput Stat Data Anal 50(4):1113–1130
33. Legrand B, Chang CS, Ong SH, Neo SY, Palanisamy N (2008) Chromosome classification using dynamic time warping. Pattern Recog Lett 29(3):215–222
34. Lendasse A, Verleysen M, De Bodt E, Cottrell M, Grgoire P (1998) Forecasting time-series by Kohonen classification. In: Proceedings of European symposium on artificial neural networks, pp 221–226
35. Lesh N, Zaki MJ, Ogihara M (1999) Mining features for sequence classification. In: Proceedings of the fifth ACM SIGKDD international conference on knowledge discovery and data mining, pp 342–346
36. Lesh N, Zaki MJ, Ogihara M (2000) Scalable feature mining for sequential data. IEEE Intell Syst Appl 15(2):48–56
37. Lessmann S, Voß S (2008) Supervised classification for decision support in customer relationship management. In: Bortfeldt A, Homberger J, Kopfer H, Pankratz G, Strangmeier R (eds) Intelligent decision support. Gabler, Wiesbaden, pp 231–253
38. Li C, Khan L, Prabhakaran B (2006) Real-time classification of variable length multi-attribute motions. Knowl Inf Syst 10(2):163–183
39. Lin SW, Chen SC, Wu WJ, Chen CH (2009) Parameter determination and feature selection for backpropagation network by particle swarm optimization. Knowl Inf Syst 21(2):249–266
40. Murzin AG, Brenner SE, Hubbard T, Chothia C (1995) SCOP: a structural classification of proteins database for the investigation of sequences and structures. J Mol Biol 247(4):536–540
41. Nanopoulos A, Alcock R, Manolopoulos Y (2001) Feature-based classification of time-series data. In: Nikos M, Stavros DN (eds) Information processing and technology. Nova Science Publishers, New York, pp 49–61
42. Ngai EWT, Xiu L, Chau DCK (2009) Application of data mining techniques in customer relationship management: a literature review and classification. Expert Syst Appl 36(2 PART 2):2592–2602
43. Pei J, Han J, Mortazavi-Asl B, Pinto H, Chen Q, Dayal U, Hsu MC (2001) PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern growth. In: Proceedings of international conference on data engineering, pp 215–224
44. Peng T, Zuo W, He F (2008) SVM based adaptive learning method for text classification from positive and unlabeled documents. Knowl Inf Syst 16(3):281–301
45. Prinzie A, Van Den Poel D (2005) Constrained optimization of data-mining problems to improve model performance: a direct-marketing application. Expert Syst Appl 29(3):630–640

46. Shi Y, Eberhart R (1998) Modified particle swarm optimizer. In: Proceedings of IEEE international conference on evolutionary computation, Anchorage, AK, USA, pp 69–73
47. Teo TSH, Devadoss P, Pan SL (2006) Towards a holistic perspective of customer relationship management (CRM) implementation: a case study of the housing and development board, Singapore. Decis Support Syst 42(3):1613–1627
48. Tsai CY, Lo CC, Lin CW (2011) A time-interval sequential pattern change detection method. Int J Inf Tech Deci Marking 10(1):83–108
49. Tsai CY, Shieh YC (2009) A change detection method for sequential patterns. Decis Support Syst 46(2):501–511
50. Tsai CY, Chiu CC (2004) A purchase-based market segmentation methodology. Expert Syst Appl 27(2):265–276
51. Tseng VS, Lee CH (2005) CBS: a new classification method by using sequential patterns. In: Proceedings of the SIAM international data mining conference, California, USA
52. Tseng VS, Lee CH (2009) Effective temporal data classification by integrating sequential pattern mining and probabilistic induction. Expert Syst Appl 36(5):9524–9532
53. Viaene S, Baesens B, Van Gestel T, Suykens JAK, Van Den Poel D, Vanthienen J, De Moor B, Dedene G (2001) Knowledge discovery in a direct marketing case using least squares support vector machines. Int J Intell Syst 16(9):1023–1036
54. Vigna G, Valeur F, Kemmerer R (2003) Designing and implementing a family of intrusion detection systems. SIGSOFT Softw Eng Notes 28(5):88–97
55. Wang FY, Carley KM, Zeng D, Mao WJ (2007) Social computing: from social informatics to social intelligence. IEEE Intell Syst 22(2):79–83
56. Wang T, Yang J (2010) A heuristic method for learning Bayesian networks using discrete particle swarm optimization. Knowl Inf Syst 24(2):269–281
57. Xing D, Girolami M (2007) Employing latent dirichlet allocation for fraud detection in telecommunications. Pattern Recog Lett 28(13):1727–1734
58. Xi X, Keogh E, Shelton C, Wei L, Ratanamahatana CA (2006) Fast time series classification using numerosity reduction. In: Proceedings of the 23rd international conference on machine learning (ICML'06), New York, USA
59. Xing Z, Pei J, Yu PS (2012) Early classification on time series. Knowl Inf Syst 31(1):105–127
60. Yang Y, Cao L, Liu L (2010) Time-sensitive feature mining for temporal sequence classification. Lect Notes Comput Sci 6230:315–326

Author Biographies

Chieh-Yuan Tsai is a professor in the Department of Industrial Engineering and Management at Yuan-Ze University, Taiwan. He received his M.S. and Ph.D. degrees in Department of Industrial and Manufacturing Systems Engineering from the University of Missouri-Columbia, USA. His research activities include data mining, customer relationship management (CRM), RFID technology and applications, and product data management.

Chih-Jung Chen is a Ph.D. candidate in the Department of Industrial Engineering and Management at Yuan Ze University, Taiwan. He received his master degree at the same institute. His research activities include data mining, sequence classification, heuristic algorithms, product development process, and module product design.

Chun-Ju Chien is a Staff Engineer of Supplier Quality Engineering Department at Showa Denko HD Trace Corp. She graduated with a degree in Department of Industrial Engineering and Management, Yuan Ze University, Taiwan. Her research interests include sequential pattern mining, sequence classification, heuristic algorithms, and customer relationship management.