International Journal of Innovative
Computing, Information and Control
Volume **7**, Number **10**, October **2011**

ICIC International ©2011 ISSN 1349-4198
pp. 5925–5938

PTCR-MINER: AN EFFECTIVE RULE-BASED CLASSIFIER ON MULTIVARIATE TEMPORAL DATA CLASSIFICATION

Chao-Hui Lee and Vincent S. Tseng*

Department of Computer Science and Information Engineering
National Cheng Kung University
No. 1, University Road, Tainan 701, Taiwan
*Corresponding author: tsengsm@mail.ncku.edu.tw

Received May 2010; revised September 2010

Abstract. *Multivariate temporal data are hybrid data. Numeric and categorical data type could be consisted of. Most past researches cannot be operated directly on the multivariate temporal data with both types. Additionally, no useful and readable rules are provided in their methods for advanced classification analysis. We proposed Progressive Temporal Class Rule Miner (PTCR-Miner) algorithm to achieve the classification on multivariate temporal data with a rule-based designed. Through our algorithm, all really useful classification rules are discovered. The rules follow the purification concept we defined, which makes rules comprehensible and intuitive for general users on data classification. We did several experiments to evaluate our method with a multivariate temporal data simulator. Experimental results show PTCR-Miner performs effectively and efficiently on the different simulated multivariate temporal datasets. That means the discovered rules are really helpful and comprehensible for data classification. Furthermore, the rule-based and flexible architecture enables PTCR-Miner more applicable to different areas of multivariate temporal data classification.*
Keywords: Rule-based, Progressive, Multivariate temporal data, Classification, Data mining

1. **Introduction.** Useful information discovery is a main purpose of data mining. With the complexity of recorded data, different kinds of data mining methods are proposed continuously. Multivariate temporal dataset is a popular recorded format recently, which describes the states of an event using different variables with the time. The completeness of these data type is preferred in many study areas, such as weather data and medical data. For classification of multivariate temporal data, there are relatively rare suitable methods and it is due to the complexity of the data type. Many temporal datasets are hybrid data, which contain categorical and numeric values. Most data mining methods focus only on numerical data or only on categorical data and their modules or methods cannot apply to this kind of dataset appropriately. Additionally, the datasets recorded in multivariate temporal mode are generally significant for advanced analyses or diagnoses. Therefore, the supply of the information about classification is important for the studies on multivariate temporal data. Many temporal related researches focused on the time series data which consisted of only numeric values and performed high accurate classification results. Several different studies applied similar concepts to multivariate time series data which consisted of multiple numeric time series data. The datasets are regarded as a matrix and transformed into lower dimension format for easier similar measurements [23,24]. However, many temporal variables are recorded in categorical type and even in different sample rate for a multivariate temporal dataset. Thus, a dataset with many

temporal variables is very different from the multivariate time series datasets and cannot be processed directly with their approaches.

The learning behavior of human being is a key point of machine learning. For classification, people always expect simple and easily comprehensible rules. The more matching descriptions make the target close the result of the rule intuitively, such as rules for sweet fruit selection and rules for disease diagnosis. Therefore, if we want to generate a human readable rule, it should not only keep the description simple and clear, but also be intuitive. In this regard, decision tree [17] is a very successful classification algorithm. It represents all classification rules in tree architecture and data are closer to a classification result, when it is matched to a deeper path of the tree. For temporal data mining, progressive confident mining [28] inherits this concept. Each discovered rule describes an event chain for one class. When an unknown-class event sequence S follows the event chain of a progressive confident rule gradually, the class of S has high probability to be the class of the rule.

The purpose of this paper is to overcome above problem to build a multivariate temporal data classifier. Besides classification, the classifier could provide classification rules and comprehensible causes and related information of each classification for users. Hence, we proposed the purification concept to enhance discovered multivariate temporal rules for classification accuracy and users comprehension. Based on the purification concept, we designed an algorithm named *Progressive Temporal Class Rule Miner (PTCR-Miner)*, for multivariate temporal data classification. The algorithm extracts the classifiable value sequences or feature sequences following the purification concept build a classifier. Each sequence keeps its target class individually. Therefore, with matching more elements of a sequence, class of a multivariate temporal instance can be recognized more obviously as the target class of the sequence. Even matching more sequences with the same target class makes classification result of a multivariate temporal instance clear and comprehensible.

Besides intuitive rules for classification, the accuracy of our method is also proven great through experimental evaluation. K-nearest neighbor algorithm for classification is a popular and high accurate mechanism. We compared our method with this classic classification mechanism. In experiments, a multivariate temporal data generator is designed to simulate possible datasets with different conditions for several performance evaluations. The classifier of *PTCR-Miner* performs better on accuracy and execution time than a KNN-based classifier in experimental results. Therefore, we infer that the purification concept does really work in our method to enhance class feature discovery on multivariate temporal datasets.

The rest of this paper is organized as follows: in Section 2, several related research works are discussed; subsequently, the problem definition and the proposed method are described in detail in Section 3; Section 4 shows all experimental results to evaluate our method; finally, all of this paper is concluded in Section 5.

2. **Related Work.** The main goal of data mining is to discover useful information. For information discovery, efficient machine learning methods and algorithms are designed in data mining study area [1,2,4,12,15,16,20,26,27]. Time series is a special data type and is one kind of temporal data. Few studies contributed practical mining method on temporal data [6,9,13]. Nevertheless, many traditional classification methods are not suitable to be applied directly. Most of time series data classification methods are based on statistical models, neural network [8,14], feature-based [5] or similarity-based techniques [25]. However, these similarity based methods and neural networks can not perform well on categorical temporal data classification.

In the past studies [18,19], KNN is proven a good and accurate classification policy when a suitable similarity measurement is chosen on time series data. Although it performs classification without any module training, but large time cost of each classification is its main drawback. In the research [19], the execution time had been improved. The speed of KNN is accelerated and the high accuracy is also kept simultaneously. In multivariate time series classification, most past researches built classifiers with mathematical models as preprocessing, such as SVD (Singular Value Decomposition) [23] and LPP (Locality Preserving Projections) [24]. In these researches, KNN is generally applied as a final classification policy. Although these methods perform well on multivariate time series data classification, their feature selection causes users more difficult to seek the possible reasons of each classification. Moreover, many dataset recorded over time are not numeric, and the importance of each temporal variable cannot be judged directly by the basic statistical information. KNN-based methods cannot be performed directly on multivariate temporal data classification.

Among the data mining studies, sequential pattern is a suitable pattern for temporal data. There exists many issues related to sequential mining [12,15,16,26]. These studies make sequential pattern more efficient and effective for different data mining techniques. These mining sequential pattern techniques are less applied on multivariate temporal sequence analyzing. In 1998, Lesh et al. [7] proposed a policy to classify temporal database in sequential patterns. The asthma related study [22] integrated similar concept and CBA algorithm for multivariate temporal data classification well. Helen et al. [16] proposed multidimensional sequential pattern mining method. Its architecture enables to be applied on multivariate temporal data. However, the defined patterns must involve all variables and this setting removes many possibly significant features of the data. The study [27] adopted sequential pattern as classification features on time series dataset, but it is still not suitable for multivariate temporal data. Besides, several methods based on sequential pattern mining offer users useful patterns as hidden important information, such as progressive confident patterns [28], change patterns [3] and surprising patterns [6]. In which increasing class confidence is considered in progressive confident pattern mining and the concept is similar the purity of decision tree. The accuracy of classification is proven to be enhanced with this concept. However, there are few methods using this property to build a multivariate temporal data classifier.

For the rule-based classifier, statistical features of useful rules are generally discussed and evaluated. Frequency threshold is a usual setting for importance of discovered items in data mining yet it is not significant enough for data classification. Many past studies [10,11,21] pointed out that confidence is the most important property of a classification rule, which consists of descriptions and an implied class label.

3. **Proposed Method: PTCR-Miner.**

In this paper, we proposed a classification method on multivariate temporal dataset. Confidence is adopted as an important indicator of our classification mining. We considered the concepts about the class purification of decision tree mining and the increasing confidence of progressive confident pattern to extract features from datasets. Following these concepts, redundant features are reduced and each discovered rule is meaningful for data classification.

3.1. **Problem definition.**

Consider a multivariate temporal dataset D. Each instance d of D consists of a multivariate temporal instance and its class label c. The whole dataset D contains m classes, and their class labels are denoted as $\{c_1, c_2, c_3, \ldots, c_m\}$. In the dataset D, each instance records e temporal variables in a period of time, and these temporal variables are denoted as $\{F_1, F_2, F_3, \ldots, F_e\}$. Thus, each instance consists of e temporal

sequences which represent recorded value series of e temporal variables respectively. We defined a temporal sequence of F_e as $\mathrm{T} = \{F_e v_1, F_e v_2, \ldots, F_e v_t, \ldots, F_e v_n\}$. The value $F_e v_t$ represents the value of F_e at time point t. Each recorded value of temporal variables may be numerical or categorical. In general case, we assume each temporal variable keeps the same sample rate, that is, each time interval between t to $(t+1)$ is the same length. As temporal variable assumption, the dataset D can also be denoted as $\mathrm{D} = \{D_k | 1 \leq k \leq e\}$. D_k is a temporal dataset, which consists of the temporal data sequences of the k^{th} temporal variables of all instances in D. In a practical case, these temporal variables may be different measurements for one object need to be described, such as many bio-signals for a patient, daily climate measurements for the atmosphere of a city.

3.2. **PTCR-Miner.** In this paper, we proposed *Progressive Temporal Class Rule Miner (PTCR-Miner)*, to achieve multivariate temporal data classification. *PTCR-Miner* is a rule-based classifier. All its rules are generated from the features of each temporal variable and the relations between the features, which associate with the classes of data.

3.2.1. *The concept of PTCR-Miner.* In a multivariate temporal dataset, each data is composed by sequences of different temporal variables. In order to reduce the dimension and complexity of data, symbolization of data value is required in data preprocessing. In this paper, we assume each temporal variable keeps its own class related features. These features and the relations between the feature set of different variables would be key points for classification. The amplitude of each temporal variable is determined by the nature of the variable and it should be not affected by other variables theoretically. Meanwhile, the amplitude of a temporal variable is usually not directly related to data classification. So, we separate variable data firstly before symbolization. As regards the discretization methods, users can choose suitable one for different temporal variable or the ones suggested by experts, such as PAA and SAX [9].

According to many classification studies, confidence of rules in a rule-based classifier is a very important attribute for accurate classification. In our method design, we not only add it in scoring policy of classification, but also put it into class rule mining as an important threshold. That is expected to makes the accuracy results of our method better and each rule in our classifier more comprehensible.

For multiple class data classification, most features in classifier building usually support more than one class. The importance of a feature for classification can not be evaluated only with its confidence values easily. Thus, we take confidence concept and classification related factors into consideration and integrate them into a "purification" concept. The purpose of purification is to extract really useful features and to make the rules generated by the features simpler and stronger for classification. The major definition of purification is to seek less class support and higher class confidence of the generated rules. For example, a generated classification rule R supports a class set $\{c_1, c_3, c_6\}$, that means that the data predicted by R are possible to be classified as one of the class set. Given another rule R' which is extended from R has a class set c_1, c_3, that is the descriptions of R' totally cover the descriptions of R. So, the confidence values of c_1 and c_3 of R' must be higher than those of R. We say that this extension obey our purification concept. R' is simpler and stronger than R for classification.

For convenience of algorithm description, few symbols and words should be defined. In our method, we expect to discover classifiable rules from the dataset D for classification. We define each discovered rule as a *Progressive Temporal Class Rule* (PTCR), which consists of one temporal sequence and a supported class set. The temporal sequence is defined as *Progressive Temporal Sequence* (PTS) or *Multivariate Progressive Temporal Sequence* (M-PTS). A PTS is formed by a sequence, $\{v_{r1}, v_{r2}, \ldots, v_{ra}, v_{rb}, \ldots, v_m | a \leq b, 1 \leq ra \leq$

$rb \leq rn$ and $a, b \in$Z}, of single temporal variable. One M-PTS is composed by time ordered PTSs of different temporal variables and we defined it as $\{PTS_{s1}, PTS_{s2}, \ldots, PTS_{sa}, PTS_{sb}, \ldots, PTS_{sn} |\ a \leq b, 1 \leq ra \leq rb \leq rn$ and $a, b \in$Z}. Besides, the supported class set is generally the subset of the class set of dataset D, which are highly supported by PTS or M-PTS. Therefore, a PTCR can be denoted as PTS: {a sub-classset of dataset D} or M-PTS: {a sub-classset of dataset D}.

3.2.2. *The algorithm of PTCR-Miner.* We integrate the key points mentioned above to propose an algorithm, namely *Progressive Temporal Class Rule Miner (PTCR-Miner)* for multivariate temporal data classification. Through the algorithm class rules of a multivariate temporal dataset can be discovered and a classification mechanism can be built by the class rules indirectly. Rule generation part is designed as an apriori-like architecture and following the purification concept.

```
Algorithm Progressive Temporal Class Rule Miner
Input : a multivariate temporal dataset (D),
        minimum support (min_sup),
        minimum confidence (min_conf)
Output : A Progressive Temporal Class Rule Set (PTCR_set)
Procedure PTCR - Miner(D, min_sup, min_conf)
1 PTCR = φ
2 For each variable v ∈ D.temporal_variables
3     set D_v = data collection of temporal variable v in D
//Progressive Temporal Sequence Mining
4     PTS_v = PTS - Mining(D_v, min_sup, min_conf)
5     D'_v = FeatureSeries(D_v, PTS_v)
//Transform each temporal sequence into a chain of PTS items
6     PTCR = PTCR ⋃ PTS_v
7 End For
8 D' = Σ_v^{#_of(D.temporal_variables)} D'_v
9 M - PTS = PTS - Mining(D', min_sup, min_conf)
10 PTCR = PTCR ⋃ M - PTS
11 return PTCR
End
Procedure FeatureSeries(D_v, PTS_v)
12 set D'_v = null
13 For each data d ∈ D_v
14    set d' = the elements of PTS_v contained by d
15    sort elements in d' with their ocurrent time
16    D'_v = D'_v ⋃ d'
17 End For
18 return D'_v
End
```

FIGURE 1. Progressive temporal class rule miner algorithm (PTCR-Miner)

Figure 1 is the algorithm of *PTCR-Miner*. The algorithm needs three input parameters, which are a multivariate temporal dataset, minimum support and minimum confidence. The output is the discovered Progressive Temporal Class Rule set of the input dataset. As the descriptions of the algorithm, the whole input dataset are separated into several sub-dataset according to temporal variables of the dataset firstly. Each sub-dataset represents

one temporal variable data in the whole multivariate temporal dataset. In general, each data instance in a multivariate temporal dataset consists of many sequences, which are recorded for different temporal variables. For example, a weather dataset is a multivariate temporal dataset, and temperature, humidity and pressure are its temporal variables. Subsequently, discretization is performed to transform data values into meaningful symbols if the data type of any sub-dataset is numeric. The data values of other categorical sub-datasets also can be processed into readable symbols if it is necessary.

```
Algorithm Progressive Temporal Sequence Mining
Input : a temporal dataset (D_t),
        minimum support(min_sup),
        minimum confidence (min_conf)
Output : Progressite Temporal Sequence(PTS)
Procedure PTS - Mining(D_t, min_sup, min_conf)
//Frequent and confident items discovery
1  FS_1 = discover_essential_item(D_t, min_sup, min_conf)
2  PTS = FS_1
3  lef T be a prefix tree built by FS_1
4  set L = 2 //depth of tree, root(L = 0), FS_1 = Root.children(L = 1)
5  do loop
//Extend the prefix tree
6     For each node p ∈ T.leaves
7        set new_nodes = {p_N | p.parent_node = p_N.parent_node and (p_N.class_set ⋂ p.class_set) ≠ null}
8        create new_nodes as p.children
9     End For
//Counting
10    For each temporal data d_t ∈ D_t
11       For each node p ∈ T.leaves
12          if (the sequence represents T.root → p is a subsequence of d_t)
13             p.support + 1
14             p.class[d_t.class].conf + 1
15       End For
16    End For
17    For each node p ∈ T.leaves
18       For each class g ∈ p.class_set
19          if ((g ∉ p.parent.class_set) or (p.class[g].conf < p.parent.class[g].conf))
20             remove g
21       End For
22       if (p.support < min_sup) or (p.class_set = φ)
23          remove p from T
24    End For
25    remove the subtrees of T without any node in layer L
26    FS_L = all_path(T)
27    PTS = PTS ⋃ FS_L
28 until T.root = null
29 return PTS
End
```

FIGURE 2. Algorithm of progressive temporal sequence mining (PTS-Mining)

3.2.3. *The algorithm of PTS-Mining.* Progressive temporal sequence mining is major part of our method. It is performed twice for mining PTS set and M-PTS set in PTCR-Miner. After data values are preprocessed, the progressive temporal sequence mining (PTS-Mining) is performed on each sub-dataset to extract progressive temporal sequences (PTS) for each temporal variable. All discovered PTSs are not only frequent sequences but also keeping with the purification concept. Figure 2 shows the algorithm of progressive temporal sequence mining in detail. We utilize the combination of branches of prefix tree architecture to handle sequence generation. Following the principle of apriori-like mechanism, the support values and confidence values for each new leaves are checked when new level of the tree is generated. Each path of the tree represents one PTS if its leaf node passes the set minimum support and follows our purification concept. With the deeper extension of the tree architecture, the longer candidate PTSs can be formed.

At the first of PTS-Mining, the items which pass minimum support and minimum confidence thresholds are discovered (line 1). Subsequently, minimum confidence setting is no longer used. We designed some tricks for keeping frequency and purification of discovered PTSs. When k-length candidate pattern generation is executed, the depth of all paths of the tree must be $k-1$. According to our tree architecture mining, k-length candidate PTSs can be absolutely generated by extension of all leaf nodes in the $k-1$ depth tree. For each leaf node, the extension is to add the elements of itself and the elements of all its sibling nodes which have the same parent node with it as its child nodes. However, the intersection of the class sets of the node and the extended one must be not null in order to obey the concept of purification. Meanwhile, each confidence value of the class sets of the extended nodes must be higher than that of their parent nodes after the whole dataset counting is completed.

3.2.4. *M-PTS Mining.* After all PTSs of each variable sub-dataset are prepared, the advanced relations between the PTSs of different temporal variables could be generated. All discovered PTSs represent important features of their own temporal variables. However, there must be useful class information hidden in relations between different temporal variables. Therefore, we translate the whole original multivariate temporal dataset into a new PTS sequence dataset. In the new dataset, each data consists of the PTSs it contains and all PTSs are ordered by their happen time. This translation makes the progressive temporal sequence mining can be performed directly on the dataset. New progressive temporal sequences consist of feature sequences instead of value sequences and they represent classifiable relations between different temporal variables. We name these relations as multivariate progressive temporal sequences (M-PTS).

After twice PTS-Mining processes, all class related features of the multivariate temporal dataset are extracted, which consists of PTSs and M-PTSs. Each feature consists of a sequence of values or PTSs and a class set they support. Following the purification definition, so the confidence values of the class set of these features are very significant. The higher confidence of a class Cy in a feature F represents that the data contains F has higher probability to belong to the class Cy. Therefore, we set directly the confidence values of class set of features as scores for temporal data classification. We name these features with the set score as "*progressive temporal class rules*" (PTCR).

3.3. **Classification with PTCR-Miner.** In following segments, we explain how to classify a class unknown multivariate temporal data in detail with PTCR classifier. Figure 3 shows the detail algorithm of classification with the discovered PTCRs. For a class unknown multivariate temporal data X, we must check which PTCR it contains. The PTCR consists of PTSs and M-PTSs, so all PTSs and M-PTSs must be checked. The contained PTSs can be discovered by tracing a reconstructed tree structure of all PTSs,

instead of brute-force matching, and M-PTSs can also be check in the same approach. Subsequently, the sum of the score for each class can be accumulated by the score of all matched PTSs and M-PTSs. The classification result is the class with the highest score.

```
Algorithm PTCR - classifier
Input : progressive temporal class rule set(PTCR_set),
        a class - unknown multivariate temporal instance(x)
Output : a classification result of input data(x.class)
Procedure PTCR - classifier(PTCR_set, x)
1  set class_set = PTCR_set.class_set
2  set a matrix class_score[size_of(class_set)] and all values as 0
3  For each rule r ∈ PTCR_set
4     if (x contains r)
5        For each class c ∈ r
6           class_score[c] = class_score[c] + r.class[c].confidence
7        End For
8  End For
9  set result_class_index = argmax class_score[y]
                            y
10 return class_set[result_class_index]
```

FIGURE 3. The algorithm of classification with PTCR classifier

4. **Experimental Evaluation.**

4.1. **Experimental design.**

4.1.1. *Synthetic dataset.* To evaluate performance of our proposed method, we designed a multivariate temporal dataset simulator to generate datasets with different attributes. As we define in problem definition, each multivariate temporal data instance does keep significant information about its class. Our simulator is designed under this hypothesis. Name, definition and default value of all parameters in the simulator are list in Table 1. The parameter *class_number* represents the number of classes in a simulated dataset. Parameters *variable_number*, *transaction_count*, *transaction_len* are three basic parameters of a general simulated multivariate temporal dataset. The class information is decided by seven parameters, which are *pattern_len*, *pattern_type_num*, *relations_var_num*, *relations_type_num*, *relations_num_in_one_instance*, *relations_len*, *useful_var_num*. The last parameter *mutation_rate* is relative to the number of noise in the simulated dataset.

4.1.2. *Compared method.* K-nearest neighbor (kNN) algorithm is popular for classification studies. It is a very accurate classification policy and it performs classification without any training. 1-NN is the simplest case of kNN and the instance can be classified as the class of the most similar historical data without any doubt for data classification. Xi et al. [25] applied this concept on time series classification using dynamic time warping (DTW) as similarity measurement and got very good accuracy results. However, based on its assumption, kNN must keep a huge historical dataset and users must suffer longer execution time to wait responses. Many studies emphasized how to speed up DTW and how to modify the similarity measurement on the datasets with different data formats. Since kNN-base method is proven the best policy for classification, we define an algorithm

TABLE 1. Parameters for the data simulator

parameter	**default value**
class_number	3
variable_number	3
transaction_count	10000
transaction_len	20
pattern_len	10
pattern_type_num	20
relations_var_num	3
relations_type_num	20
relations_num_in_one_instance	2
relation_len	10
useful_var_num	3
mutation_rate	0.1

Sum-1NN as a kNN-based method and attempt to make it suitable for multivariate temporal series datasets in following experiments. The detail algorithm of Sum-1NN is shown in Figure 4.

```
Algorithm Sum - 1NN
Input : A multivariate time series dataset (D),
   A class - unknown multivariate time series instance (X)
Output : the predicted class of X (class_label_of_X)
Procedure
1.  set closest_instance to null
2.  set closest_similarity to null
3.  for each instance d ∈ D
4.     if ( similarity(d, X) > closest_similarity or closest_similarity == null )
5.        closest_instance = d
6.        closest_similarity = similarity(d, X)
7.     end if
8.  end for
9.  return closest_instance.class
End
```

FIGURE 4. The algorithm of Sum-1NN

4.1.3. *Evaluation metrics.* We want to prove that the rule-based *PTCR-Miner* can provide the same or higher accuracy than kNN based methods. In all experiments, we randomly select 70% data as training part and 30% as testing part. The training part is prepared for training of our method and for historical dataset of Sum-1NN. Subsequently, performance of both methods is evaluated with the data of testing part. After all kNN-based method is difficult to process the data with categorical values in temporal datasets; we only generate numeric data in following experiments with our data simulator. In all experiments, we compare our methods *PTCR-Miner* and Sum-1NN on the accuracy and execution time of multivariate temporal data classification. Each experimental value is the average value from 10 times repeated experiments. For each repeated experiment, a

multivariate temporal dataset is re-simulated according to parameter settings. In accuracy evaluations, we regard the percentage of data which are classified correctly in testing part as a criterion. Thus, we define the criterion "accuracy" as following formula:

$$\text{Accuracy}(\%) = \frac{\text{the number of the instances which are classified correctly in testing part}}{\text{the number of instances in testing part}}$$

In execution time evaluations, we compare the processing time of building a classifier with the training part of data and evaluating the classifier with the rest testing part of data. kNN based classifier does not need training so we set the training data as its reference data to evaluate the testing data. That is, the whole execution time of Sum-1NN is spent on evaluation.

4.2. **Experimental results.**

4.2.1. *Effects of varying support and confidence on accuracy.* In following two experiments, we evaluate accuracy and execution time of our method with varying setting of *minimum support* and *minimum confidence*. We set *minimum support* = 0.05 and *minimum confidence* = 0.2 as default values. Figure 5(a) shows the accuracy results with varying support setting. The method Sum-1NN does not need parameter setting so we compare with its average accuracy. The accuracy of our method grows up with the decreasing of *minimum support*. That means that lower threshold makes our classifier more accurate. Under smaller value of *minimum support* cause more possible sequences are considered and more useful rule are discovered. Therefore, we can know the relationship between accuracy of classification results and *minimum support* value is reverse. Figure 5(b) shows the execution time of both methods. Thus, the execution time of Sum-1NN is almost a constant value for comparison with our method in Figure 5(b). Execution time of our method is increasing slightly with decreasing of *minimum support* but all its values are smaller than ones of Sum-1NN. Figure 6 shows accuracy results with varying

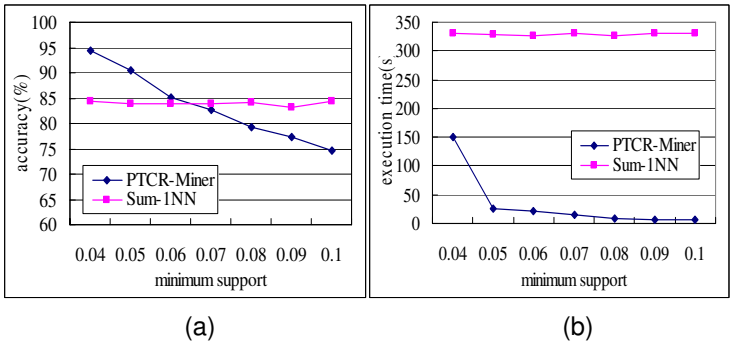

FIGURE 5. (a) *Minimum support* vs. accuracy results and (b) *minimum support* vs. execution time

minimum confidence setting of our method. As last experiment, the accuracy results and execution time of Sum-1NN are reference data to evaluate the results of our method. In Figure 6(a), our method reaches highest accuracy result at *minimum confidence* = 0.2. Basically, the lower confidence setting makes higher accuracy of our classification result but accuracy goes down when *minimum confidence* is set too small. For execution time of our method, its value grows up with reduction of *minimum confidence*. However, all execution time of our method are less than that of Sum-1NN.

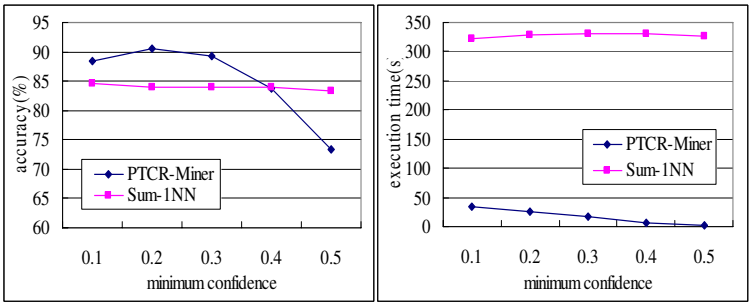

FIGURE 6. (a) *Minimum confidence* vs. accuracy results and (b) *minimum confidence* vs. execution time

4.2.2. *The number of relation types vs. accuracy.* Figure 7 shows accuracy results with varying *relation_type_num* setting of data simulator. The parameter *relation_type_num* represents the quantity of relation types and the relation is the classification information between different temporal variables. In our definition, a relation is an M-PTS. In Figure 7(a), the accuracy trends of both methods are almost the same. The accuracy of *PTCR-Miner* is always higher than the accuracy of Sum-1NN. The accuracy results of both methods go down with increasing *relation_type_num*. For a simulated dataset, that more relation types increase the number of M-PTSs. However, the unchanged number of PTSs lets all generated M-PTSs too easy to keep similar PTSs. Thus, the dataset becomes not conducive to be classified. In Figure 7(b), the frequency of each relation type becomes lower with the increasing of relation types. It causes that *PTCR-Miner* discovers fewer rules under the same mining threshold setting and costs less time on rule mining. Nevertheless, the kNN-based method is not affected for any information hidden in data on execution time.

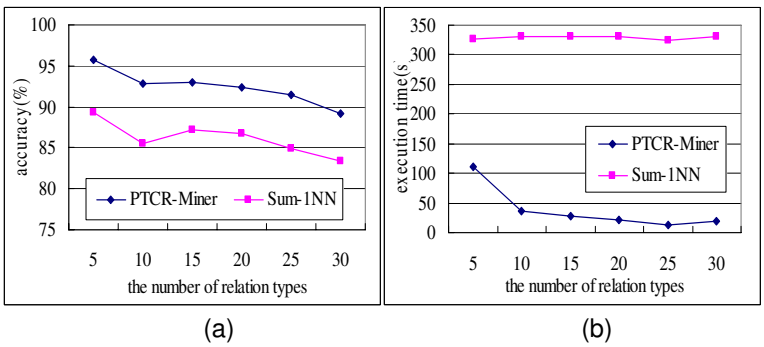

FIGURE 7. (a) Parameter *relation_type_num* vs. accuracy and (b) parameter *relation_type_num* vs. execution time

4.2.3. *The number of pattern types vs. accuracy.* Figure 8 shows accuracy results with varying *pattern_type_num* setting of data simulator. The parameter *pattern_type_num* controls the number of pattern types of each temporal variable. A pattern is a PTS in our definition. The accuracy of both methods goes down with increasing *pattern_type_num*

in a simulated dataset. The more types of class information obviously make lower accuracy results of both classifiers. With the increasing *pattern_type_num*, the quantity of pattern types of each temporal variable increases. That shrinks the support of each pattern and it indirectly reduces the characteristic of classes of the simulated dataset. That is the major reason to result in decreasing accuracy trends of both methods. In execution time shown in Figure 8(b), the Sum-1NN almost has no change with varying *pattern_type_num*. However, our method performs using less time with larger *pattern_type_num* setting. There are fewer rules can be discovered in our method, because the frequency of each pattern goes down. Simultaneously, the execution time of *PTCR-Miner* is also diminished for this situation.

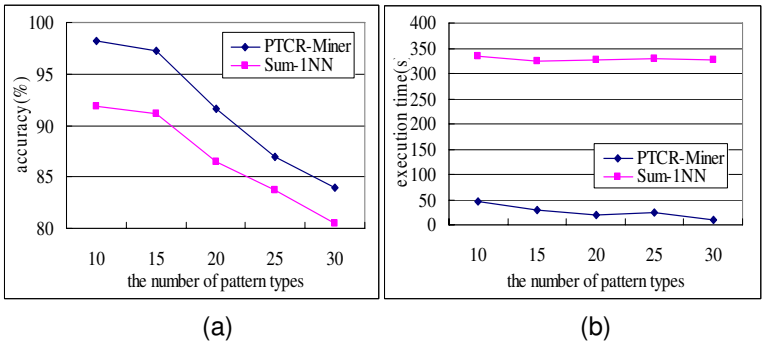

FIGURE 8. (a) Parameter *pattern_type_num* vs. accuracy and (b) parameter *pattern_type_num* vs. execution time

4.2.4. *The number of useful temporal variables vs. accuracy.* Figure 9(a) shows the accuracy results of both methods with varying *useful_var_num*. The parameter *useful_var_num* represents the number of the temporal variables related to class label in the simulated dataset. That is the number of the temporal variables which keep useful information for classification. In general cases, the class related information would be partially related to temporal variables. We changed *useful_var_num* from 3 to 6 and set *variable_number* as 6. It would be easier to observe the effect of *useful_var_num* in this experiment. According the results of Figure 9(a), *PTCR-Miner* outperformed Sum-1NN in all partial related conditions and the highest accuracy results of both methods are at the fully related condition. The accuracy trends of both methods go down with the value of *useful_var_num*. The trend of Sum-1NN almost decreases linearly. *PTCR-Miner* is not so sensitive to these changes, and its trend has a lower result at *useful_var_num*=2. The reason is that the general kNN-based methods assume all temporal variables are related to class. However, *PTCR-Miner* is designed to discover class related information so it would be affected very slightly by the redundant temporal variable. Figure 9(b) shows the execution time results of both methods. The execution time trends of both methods grow down with decreasing the redundant temporal variables. That means that redundant information causes more execution time. In whole results, *PTCR-Miner* still outperforms than Sum-1NN on execution time.

5. **Conclusions.** In this paper, we proposed progressive temporal class rule miner algorithm, which is a rule-based multivariate temporal data classification method and named *PTCR-Miner*. Meanwhile, we defined a purification concept to enhance class related information mining. According to the concept, *PTCR-Miner* discovers all class related

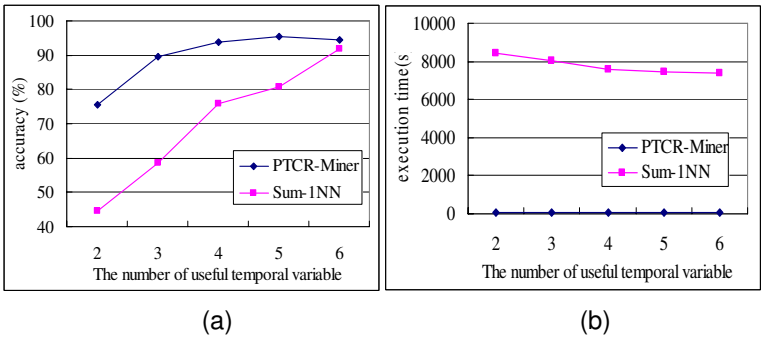

FIGURE 9. (a) Parameter *useful_var_num* vs. accuracy and (b) parameter *usefule_var_num* vs. execution time

features and the relations between the features as a classifiable rule set. A classification mechanism is established with the discovered rule set. Through experimental evaluation, *PTCR-Miner* always performed well and stably under different parameter setting of the data simulator. Rule-based design makes our classifier not only classify multivariate temporal data accurately, but also provide users the traceable reasons of data classification.

Actually, there are few deficiencies in our classification mechanism, which mainly contain parameter setting of our method and the quality of discovered rules. In the future, we will enhance it on several directions, which include free parameter setting and rule enhancement. Parameter setting of *PTCR-Miner* is intuitive for users. However, all parameter setting is always a challenge for users; we will remove parameter setting to make the setting of our method friendlier. Additionally, more class relative factors will be considered under the purification concept and rule intuition constraints. We will take the opposite rules into consideration, which keeps inverse relations between sequences and class. Those rules can apply to many condition preventions and also can enhance the classification ability of our method.

Acknowledgement. This research was supported by National Science Council, Taiwan, under Grant No. NSC 98-311-B-006-003 and NSC 97-3114-E-006-001.

REFERENCES

[1] R. Agrawal and R. Srikant, Fast algorithms for mining association rules, *Proc. of the 20th Int. Conf. on Very Large Databases*, Santiago, Chile, 1994.

[2] R. Agrawal and R. Srikant, Mining sequential patterns, *Proc. of the 11th Int. Conf. on Data Engineering*, Taipei, Taiwan, 1995.

[3] M. Botsch and J. A. Nossek, Feature selection for change detection in multivariate time series, *Proc. of IEEE Symposium on Computational Intelligence and Data Mining*, Honolulu, HI, pp.590-597, 2007.

[4] R. W. Chang, K. S. Weng and L. Yao, Clustering of incomplete data based on ellipsoids with adaptive volumes, *ICIC Express Letters*, vol.3, no.4(A), pp.1037-1042, 2009.

[5] S. Hengpraprohm and P. Chongstitvatana, Feature selection by weighted-SNR for cancer microarray data classification, *International Journal of Innovative Computing, Information and Control*, vol.5, no.12(A), pp.4627-4636, 2009.

[6] E. Keogh, S. Lonardi and W. Chiu, Finding surprising patterns in a time series database in linear time and space, *Proc. of the 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, Edmonton*, Alberta, Canada, pp.550-556, 2002.

[7] N. Lesh, M. J. Zaki and M. Ogihara, Mining features for sequence classification, *Proc. of the 5th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, San Diego, CA, pp.342-246, 1999.

[8] C. Li, L. Khan and B. Prabhakaran, Real-time classification of variable length multi-attribute motions, *Knowledge and Information Systems*, vol.10, no.2, pp.163-183, 2006.

[9] J. Lin, E. Keogh, S. Lonardi and B. Chiu, A symbolic representation of time series, with implications for streaming algorithms, *Proc. of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, San Diego, CA, pp.2-11, 2003.

[10] B. Liu, W. Hsu and Y. Ma, Integrating classification and association rule mining, *Proc. of the 4th Int. Conf. on Knowledge Discovery and Data Mining*, New York, pp.80-86, 1998.

[11] B. Liu, Y. Ma, C. K. Wong and P. S. Yu, Scoring the data using association rules, *Applied Intelligence*, vol.18, no.2, pp.119-135, 2003.

[12] H. Mannila, H. Toivonen and A. I. Verkamo, Discovery of frequent episodes in event sequences, *Data Mining and Knowledge Discovery*, vol.1, no.3, pp.259-285, 1997.

[13] Y. Matsumoto and J. Watada, Knowledge acquisition from time series data through rough sets analysis, *International Journal of Innovative Computing, Information and Control*, vol.5, no.12(B), pp.4885-4898, 2009.

[14] A. Nanopoulos, R. Alcock and Y. Manolopoulos, Feature-based classification of time-series data, *Information Processing and Technology*, pp.49-61, 2001.

[15] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal and M.-C. Hsu, PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth, *Proc. of the 17th Int. Conf. Data Engineering*, Heidelberg, Germany, pp.215-224, 2001.

[16] H. Pinto, J. Han, J. Pei, K. Wang, Q. Chen and U. Dayal, Multi-dimensional sequential pattern mining, *Proc. of the 10th Int. Conf. on Information and Knowledge Management*, Atlanta, GE, pp.81-88, 2001.

[17] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufman Publisher, San Francisco, 1993.

[18] C. A. Ratanamahatana and E. Keogh, Making time-series classification more accurate using learned constraints, *Proc. of SIAM International Conference on Data Mining*, Lake Buena Vista, FL, pp.11-22, 2004.

[19] C. A. Ratanamahatana and E. Keogh, Everything you know about dynamic time warping is wrong, *The 3rd Workshop on Mining Temporal and Sequential Data, the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Seattle, WA, 2004.

[20] M. V. Sudhamani and C. R. Venugopal, Nonparametric classification of data viz. clustering for extracting color features: An application for image retrieval, *ICIC Express Letters*, vol.1, no.1, pp.15-20, 2007.

[21] V. S. Tseng and C. H. Lee, Effective temporal data classification by integrating sequential pattern mining and probabilistic induction, *Expert System with Applications*, vol.36, no.5, pp.9524-9532, 2009.

[22] V. S. Tseng, C. H. Lee and J. C. Chen, An integrated data mining system for patient monitoring with applications on asthma care, *Proc. of the 21th IEEE International Symposium on Computer-Based Medical Systems*, Finland, pp.290-292, 2008.

[23] X. Weng and J. Shen, Classification of multivariate time series using two-dimensional singular value decomposition, *Knowledge-Based Systems*, vol.21, no.7, pp.535-539, 2008.

[24] X. Weng and J. Shen, Classification of multivariate time series using locality preserving projections, *Knowledge-Based Systems*, vol.21, no.7, pp.581-587, 2008.

[25] X. Xi, E. Keogh, C. Shelton, L. Wei and C. A. Ratanamahatana, Fast time series classification using numerosity reduction, *Proc. of the 23rd Int. Conf. on Machine Learning*, New York, pp.1033-1040, 2006.

[26] D. Xin, J. Han, X. Yan and H. Cheng, Mining compressed frequent-pattern sets, *Proc. the 31st Int. Conf. on Very Large Data Bases*, Trondheim, Norway, pp.709-720, 2005.

[27] M. J. Zaki, Efficient enumeration of frequent sequences, *Proc. the 7th Int. Conf. on Information and Knowledge Management*, Washington DC, pp.68-75, 1998.

[28] M. Zhang, W. Hsu and M. L. Lee, Mining progressive confident rules, *Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Philadelphia, PA, pp.803-808, 2006.