Categorical Time-Series Data Classification base on Sequential Pattern

Vincent Shin-Mu Tseng Chao-Hui Lee
Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, Taiwan, R.O.C.
Email: tsengsm@mail.ncku.edu.tw

Abstract *– Rich kinds of time-series data exist in wide application domains. These data, like microarry data set, are usually hard to handle with common statistical methods. Inherently, there exist interesting correlation between the time-series data itself and some associative class label. The motivation of our research is to explore the issue of data classification based on time-series data. Although a number of methods have been proposed for solving the classification problem based on the well-known learning models like decision tree or neural network, they may not perform well in mining datasets with time sequence property like time-series gene expression data. In this paper, we propose a new data mining method, namely Classify-By-Sequence (CBS), for classifying large time-series datasets. The CBS method mainly utilizes the concept of sequential pattern mining and probabilistic reasoning. We designed two policies namely CBS-Class and CBS-All for predicting the class of new data instances. Finally, we evaluate the performance of CBS in comparison with other methods through several experiments. The experiments show that CBS achieves better performance in both of accuracy and execution efficiency.*

Keywords: Sequential Pattern, Data Mining, Classification, Time Series Data

1. Introduction

In recent years, various kinds of data mining techniques have been proposed, like Association Rule, Sequential Pattern, Clustering and Classification Modeling. These techniques are also well applied in different application fields. Most of them were developed and studied independently, and compounded methods are rare. Each mining technique can extract one part of data information by partial data features. Some compound methods were developed by combining two or more data mining techniques and usually they perform better in accuracy than the other method just has one technique. E.g. classification by association rules (CBA)[9], or classify sequence data by integrating decision tree and sequential pattern features [7]. Although some studies have proposed integrated data mining techniques like in [7, 9], there exist few researches on classifying large datasets by using sequential patterns directly. This motivates the research proposed in this paper.

Recently, biotechnology develops rapidly, and microarray data has played an important role of each bio-relative research. The classification of microarray data becomes an essential data mining application. Especially, a series of microarray experiments are the most popular and complete approach for taking down bio-events. Some of them are time-series data and it is a challenge to extract the important information by using existing data mining method. Therefore, we target on the time-series data classification problem with specialization in categorical time-series data classification.

Although the existing classification methods can provide good performance and accuracy on traditional types of datasets, they are not capable of classifying the time series-related data. In [9], Bing Liu et al. proposed a method that can classify the data by using association rules, namely CBA. This approach brought up a new concept about integrating well-known techniques like association rule discovery to create a new classification method. In [9], it was reported that CBA can achieve higher accuracy than traditional classification methods like C4.5 [13]. Zaki [7] proposed a classification method that combines sequential pattern discovery and classical classification methods to classify data. Although it is also a compound method attempts to improve the classification by combining two algorithms, the concept and methodology are different from CBA. This method is to classify the data with traditional classification using sequential pattern as a preprocessing method to extract the features of data. Its experimental results showed that the algorithm perform in higher accuracy than original classification. This also indicates that utilizing the time information can improve the accuracy in classification.

In this paper, we aim at combining completely different data mining techniques to find out more data characteristics so as to improve the accuracy of

classification work. We proposed a new method named CBS that can discover the sequential rules from time series data and classify the data directly by using these rules rather than using classical classification method. The method works with easy processes, and takes more factors that could affect mining results into account than common data mining techniques do. Through experimental evaluation, our method was shown to deliver better performance than other methods.

In Section 2, we give a short definition of the targeted problem. The proposed method, CBS, is described in details in Section 3. The experimental evaluation results are given in Section 4. A conclusion is given in Section 5.

2. Problem Definition

We assume a database that stores a lot of time-series data, and these data are separated into some different classes. By learning from the time series database, we aim at building a classifier to judge which class the new data belongs to. Let D_i represents all time-series data set belonging to class i. So, the database is represented as D = {D_1, D_2, D_3, D_4, … }. Each class data set D_i is composed of time-series transactions with the form as {a_1, a_2, ...}. In this kind of transaction, a_n represents the value at one time point. If the value of each time point is continuous, we must transform them into categorical values before the algorithm. We want to find out the classifiable sequences to build a classifier. Traditional sequential pattern method cannot efficiently find out the classifiable sequences, those just represent the frequent sequences of all transactions. We consider the class label of data in the sequence mining procedure, and try to find out the CSPs to be classifier rules. The CSP rule is in the form of sequence with class information, like $SP_i \rightarrow C_m$, where SP is a sequence like $a_2 \rightarrow a_3 \rightarrow a_7$…, and if the subsequence of the sequence x equal to the SP of CSP y, we call that the sequence x matches CSP y. Hence, for classifying a new time-series data, we use the class information of the CSPs which match the sequence to judge which class the data should belong to.

3. CBS Method

We propose two policies to discover these classifiable sequences. The next couple paragraphs will describe these two methods in details.

CBS_ALL

We try to extract all classifiable sequences from whole time-series database. The concept of this method is similar to the sequential pattern mining, but it is more complicated than that. Besides finding out the frequent sequences, we also need to judge the classify-score of these sequences. CBS_ALL algorithm considers the class support and transaction support at the same time. We assume an Apriori-like procedure, and implement the most important processes into the part of support counting. Figure 1 shows the whole CBS_ALL algorithm in details. After mining, we can get a lot of CSP rules (Classifiable Sequential Pattern). Each CSP rule contains the classification information, and we need to use all CSP rules to build a classifier. Figure 2 describes a usage policy, which express how to use all CSP rules to classify a time-series data correctly.

As mentioned in problem definition, this algorithm can just process the categorical tmie-series data. Therefore before mining, we must transform the all transaction data into category sequence format. In algorithm we extract large-1 items as CSP_1 , and then we use CSP_1 to generate SP_2 (candidate 2 Sequential Pattern). In our method, we use both sequence support value and class support value of a SP to determine if this SP can become a CSP. SP.seq_sup (sequence support) shows the number of the transactions matching SP. But we define the class support with another policy. We count it separately for each class data. SP.class_sup[x] (class support) represents the number of transaction matching SP with class label x. In the algorithm, we process both counting operation at the same time. After whole dataset is counted for SP, we prune SP into CSP with their seq_sup and class_sup[]. A CSP must be a SP with SP.seq_sup larger than min_seq_sup and one of SP.class_sup[] larger than min_rule_sup. The pruning of min_rule_support means this rule must have at least one class support larger than minimum rule support, else it represents this sequential pattern distributes in almost all classes without ability to be a rule. Then the procedure goes back to candidate generation. The algorithm looply generatesall CSP

```
CBS _ ALL (Dataset D, min_seq_su p, min_rule_s up)
{
  CSP_1 = {large 1-items}
    for(i = 2; CSP_i ≠ φ; i++) do
      SP_i = gen _ candidateS P(CSP_{i-1});
      for each data d ∈ D do
        SP_s = SP_i ∩ subseq(d);
        for each sp ∈ SP_s do
          sp.seq_ sup++;
          sp.class_ sup[d.class]++;
        end
      end
      CSP_i = {sp|sp ∈ SP_i, sp.seq_ sup ≥ min_seq_su p
and ∃w let sp.class_ sup[w] / sp.seq_ sup ≥ min_rule_s up}
    end
  CSP = ∪_i CSP_i
}
```

Figure1. CBS ALL algorithm

until no more SP can pass the min_seq_sup and min_class_sup thresholds.

```
Class _of_ _sequence (sequence x)
{
   M = φ;
   for each csp_i ∈ CSP do
      if csp_i.sp ∈ subseq (x)
         M.add(csp_i);
   end
   score_arra y[] = new array[clas s_set(D).c ount];
   for each csp_m ∈ M do
      for each c_n ∈ class _set (D) do
         score_arra y[n] + =
csp_m. support / csp_m.class_sup [n];
      end
   end
   k = indexof (Max{score_ array[]});
   return k;
}
```

Figure2. CBS ALL classifier

CBS_ALL classifier

This algorithm is the guideline for deciding how to use CSPs to classify new time-series data. In this procedure, we use scoring method to evaluate which class the new sequence should belong to. Figure 2 shows the procedure in details.

We find out all CSP with sequence data belonging to the subsequence of the new sequence. We use the class support and sequence support to calculate the all class scores of each selected CSP. Finally, we count the scores for each class with probability form, and we classify the new sequence into the class that gets the highest score. Obviously, this is a simple process for counting score to classify a new sequence data, but the algorithm has already considered the two important factors – the length of CSP and the subsequences CSP of the matched CSP. As an easy example, suppose a CSP A is the subsequence of CSP B. Hence, if a new sequence contains CSP B it must also contain CSP A. It means we count some scores twice. For this reason, it seems that we should remove all matched sequences that are contained by other longer CSPs, like CSP A. But another factor – the length of the matched CSP eliminates this problem. According to the scoring study, we need to weight CSP with their sequence length. Otherwise it means we define length 5 CSP to get the same level with length 1 CSP. And how can we weight the CSP with its sequence length? We have tried product of the length and the original score, but it does not return a good answer. In conclusion, we use the subsequence relation to weight the CSP. We don't remove all it subsequences score counting, then it is weighted directly and easily. So we simplify the algorithm and solve these two problems.

CBS_CLASS

In this part we introduce another policy of CBS. This algorithm has higher time complexity, but it is more stable and has higher accuracy for different status dataset. This method separates the database into groups by class label. We retrieve all classifiable sequences from each class group. We build the classifier by these sequences. The concept of this method focuses on the features of each class group. We classify a new sequence with features retrieved from each class, instead of features of whole dataset. This concept is more effective than CBS_ALL. For classification study, if we want to build a classifier, we must know the rules of each class. So getting the rules from each class dataset is more correctly than mining them from whole database. Figure 3 shows the detail of the CBS_CLASS algorithm.

```
CBS_CLASS( Dataset D, min_sup)
{
   for each c_i ∈ class_set( D) do
      D_i = class _dataset (D, c_i);
      CSP_i = FindSP( D_i, min _ sup )
}
FindSP(Dat aset D, min_sup)
{
   SP_1 = {large 1 - items}
   for( i = 2; SP_{i-1} ≠ φ; i + +) do
      SP_C_i = gen_candid ateSP( SP_{i-1});
      for each data d ∈ D do
         SP_s = SP_C_i ∩ subseq(d);
         for each sp ∈ SP_s do
            sp.sup + +;
         end
      end
      SP_i = {sp | sp ∈ SP_s, sp.sup ≥ min_sup}
   end
   return ⋃_i SP_i
}
```

Figure3. CBS CLASS algorithm

It is different from CBS_ALL algorithm. CBS_CLASS just needs dataset and the parameter - minimum support. FindSP in Figure 3 is an Apriori-like sequential pattern mining procedure. After sequential pattern extraction for each class, we can use these sequential patterns to classify the new sequence data directly.

CBS_CLASS classifier

We show the classifier algorithm in Figure 4. This algorithm is also designed to classify new sequence with scoring method. But there is no sequence support and class support scores in CBS_CLASS. So we must set the class score for each CSP. As the process is represented in CBS_CLASS classifier algorithm, we use the

sequence length to be the CSP core. We also normalize the total score of each class into the same value. The maximum score for a sequence for each class is 1. According to the experiments, it seems not possible that two classes got the same score.

This method is more reasonable on sequence feature mining. It not only indirectly eliminates the data quantity unbalance factor between class datasets, but also extracts the real sequential pattern for each class sequence data. It is different from CBS_ALL classifier. The CSPs of CBS_ALL are frequent sequence of whole dataset, so they may be just general features for time-series data and not good features for classification.

```
class_of_s equence(se quence x)
{
  total_ score[] = new array[clas s_count(D) ];
  for each csp_i ∈ CSP do
    to tal_score[ csp_i.class] + = csp_i.sp.length ;
  end
  score_arra y[] = new array[clas s_count(D) ];
  for each csp_i ∈ CSP do
    if csp_i ∈ subseq(x)
      for each c_m ∈ belong_cla ss_set( csp_i )
        score_arra y[m] + =
csp_i.sp.length / total_s core[m];
      end
    end if
  end
  k = index_of(M ax{score_a rray[]});
  return c_k
}
```

Figure4. CBS CLASS classifier

4. Experimental Evaluation

We designed a simulator to weave the time-series data. We can make many different experiments by changing the parameters of the simulator. In the following paragraph, we will introduce the simulator and show the experiment result of CBS_ALL and CBS_CLASS sequence classification on simulated data.

Data simulator

We design a time-series data generator that assumes the situation of data is the same as the problem definition. Hence each time-series data transaction belongs to one class, the same length, and possible with classifiable sequential pattern(s). Beside these basic features of a time-series dataset, we also consider other factors could be able to affect the classification result. We assume that there is one or more sequential patterns could be hidden in one class dataset. And for each class, there are frequent items or their own region. Even there is mutation rate for each sequence data, that can replace items randomly in sequence to simulate the outliers or some error values. Table 1 shows the main parameters used in the data simulator.

Table1. Parameters for the synthetic data generator.

Parameter	Description	Default Value
seq_len	The number of items in each time sequence	10
pattern_len	The length of hidden sequential patterns	5
value_level	The level of value	100
seq_count	The number of time sequences	5000
class_count	The number of classes	10
pattern_count	The number of hidden sequential pattern	5
item_pattern_prob	The probability of frequent items in hidden sequential patterns	0.3
item_seq_prob	The probability of frequent item in a transaction	0.3
mutation_prob	The probability of item mutation	0.25
skew_ratio	The degree of skewed distribution of sequence patterns in classes	0

After simulation parameter setting, the simulator just produces a lot of sequences of each class following the setting value. Then the simulator randomly accesses the sequencees from each class dataset to weave a real time-series data file.

Experiments

We design a series of experiments to evaluate, the perfomrance of CBS_ALL and CBS_CLASS algorithms under different types of datasets. Furthermore, we evaluate the impact of parameter variation on CBS_ALL and CBS_CLASS.

We did following experiments (Figure 5. and Figure 6.) to compare the classification ability of both algorithm. The most important part of out

experiments is to discuss the relation between algorithms' accuracy and a simulation parameter. As we know, some experimental results are common and expectable. For example, increasing mutation rate will decrease the accuracy, and when we reduce class count, the precision would be higher. Therefore in the next part, we focus on discussions on varying two main parameters that are expected to produce interesting results.

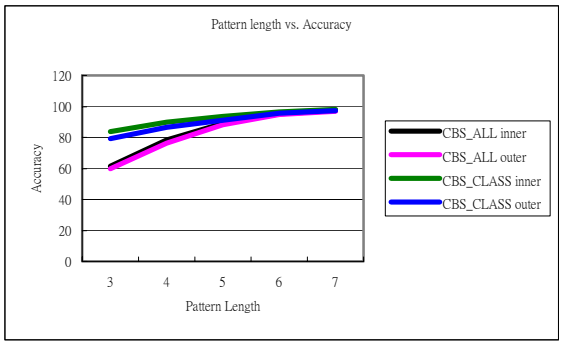

Figure 5. Pattern length and Accuracy relation map

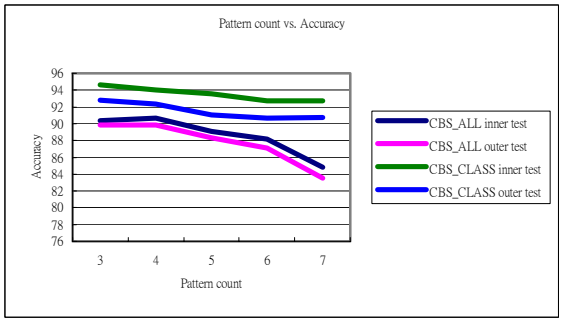

Figure 6. Pattern count and Accuracy relation map

The first experiment results are shown in Figure 5. We can see when hidden pattern is longer, the accuracy of both algorithms is higher. However, when pattern length 5 is less than 5, CBS_CLASS performs better than CBS_ALL, and the shorter the pattern length is, the difference of them is larger. This means that CBS_CLASS is easier to extract hidden pattern information than CBS_ALL, because it considers each class dataset individually and will miss less important pattern than CBS_ALL. And the pattern missing rate of CBS_ALL will be improved when hidden information becomes more obvious. Figure 6 shows the second experimental result. We also get the same observation that CBS_CLASS can get higher precision than CBS_ALL. The x-axis represents the number of hidden sequential pattern of each class. And the more hidden patterns, the time-series data of each class will become more random and hard to extract information. The result of experiment matches our inference. The accuracy of both algorithms gets down when the number of hidden pattern is increased. Therefore the result of experiment shows that the CBS_CLASS is more stable than CBS_ALL when data information hides more randomly. Comparing the concepts of both algorithms, the method of CBS_CLASS processes each class dataset make data mining procedure become more stable and powerful.

When we set the same parameters in CBS_ALL and CBS_CLASS, CBS_ALL executes faster, but CBS_CLASS always can get better result in accuracy. Hence, beside concept comparison from algorithm, we also can get the support from all experiments to prove CBS_CLASS is more ideal for sequence classification.

Finally, we designed a set of experiments to compare our method CBS_CLASS with the SPF-classifier (Sequential Pattern Feature classifier) proposed in [7]. We use C4.5 as the classifier component for SPF-classifier. The minimum support is set as 0.03. Figure 7 shows the accuracy of both algorithms in terms of inner test and outer test, with parameter *pattern_len* varied from 3 to 7. In overall, SPF-classifier delivers stable accuracy over different *pattern_len*, while CBS_CLASS presents higher accuracy with *pattern_len* increased. Moreover, SPF achieves comparable accuracy with CBS_CLASS for inner test, but it performs much worse than CBS_CLASS for outer test, especially under larger *pattern_len*.

Figure 8 is another experiment showing the accuracy of both algorithms in terms of inner test and outer test, with parameter *pattern_count* varied from 3 to 7. The average accuracy of CBS_CLASS is still higher than SPF-classifier.

Figure 9 shows the last experimental result that CBS_CLASS performs better than SPF-classifier in skewed dataset. The *skew_rate* is from 0.2 to 0.8. The result indicates that CBS_CLASS can perform well and deliver the stable performance result under skew data.

According to the results of the above three experiments for comparison of CBS with SPF-classifier, we conclude that the CBS_CLASS method can perform well than other methods proposed previously on time-series datasets classification.

5. Conclusions

We have presented two new methods for time-series data classification, which are shown to have good performance via experimental evaluation. It is shown that we can classify time-series data by using sequential patterns, and these methods are more easy and direct to utilize than other methods proposed previously. Hence, the main contribution of this paper is in constructing a new and integrated data mining approach for classifying time-series data. For future work, we will try to improve the methods so that it could also handle the continuous type time-

series dataset. Furthermore, we shall investigate the application of this method on datasets with rare classes or those with the missing values. I

ACKNOWLEDGEMENT

This research was partially supported by National Science Council, Taiwan, R.O.C., under grant number NSC91-2321-B006-003.

References

[1] R. Agrawal and R. Srikant, Fast Algorithms for Mining Association Rules, Proc. of the 20th Int'l Conference on Very Large Databases, Santiago, Chile, September 1994.

[2] R. Agrawal and R. Srikant, Mining Sequential Patterns, Proc. of the 11th Int'l Conference on Data Engineering, Taiwan, March 1995.

[3] K. Ali, S. Manganaris, R. Srikant, Partial Classification using Association Rules, Proc. of the 3rd Int'l Conference on Knowledge Discovery in Databases and Data Mining, Newport Beach, California, August, 1997.

[4] R. J. Bayardo Jr. Brute-Force Mining of High-Confidence Classification Rules. Proc. of the Third International Conference on Knowledge Discovery and Data Mining, pp. 123-126, 1997.

[5] L. Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone, Classification and Regression Trees, Wadsworth, Belmont, CA, 1984.

[6] U. M. Fayyad and K. B. Irani, Multi-interval discretization of continuous valued attributes for classification learning. In R. Bajcsy (Ed.), Proc. of the 13 International Joint Conference on Artificial Intelligence, pp. 1022-1027, Morgan Kaufmann, 1993.

[7] N. Lesh, M. J. Zaki, M. Ogihara, Mining features for Sequence Classification, 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 342-246, San Diego, CA, August 1999.

[8] W. Lee, S. J. Stolfo, K. Mok, Mining Audit Data to Build Intrusion Detection Models, Proc. KDD-98, honorable mention best application paper, August 1998.

[9] B. Liu, W. Hsu, Y. Ma, Integrating Classification and Association Rule Mining, Proc. of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98, full paper), New York, USA, 1998.

[10] B. Liu, W. Hsu and S. Chen, Using General Impressions to Analyze Discovered Classification Rules, Proc. of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97, full paper), pp. 31-36, August 14-17, Newport Beach, California, USA, 1997.

[11] B. Liu, W. Hsu, Y. Ma, Mining Association Rules with Multiple Minimum Supports, Proc. of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD-99), August 15-18, San Diego, CA, USA, 1999.

[12] M. Mehta, R. Agrawal and J. Rissanen, SLIQ: A Fast Scalable Classifier for Data Mining, Proc. of the Fifth Int'l Conference on Extending Database Technulogy, Avignon, France, March 1996.

[13] J. R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufman, 1992.

[14] J. C. Schlimmer, Efficiently Inducing Determinations: A Complete and Systematic Search Algorithm that Uses Optimal Pruning. International Conference on Machine Learning, pp. 284-290, 1993.

[15] R. Srikant and R. Agrawal, Mining Quantitative Association Rules in Large Relational Tables, Proc. of the ACM SIGMOD Conference on Management of Data, Montreal, Canada, June 1996.

[16] K. Wang, W. Tay, B. Liu, An Interestingness-Based Interval Merger for Numeric Association Rules, Proc. of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98), New York, USA, 1998.

[17] K. Yoda, T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, Computing Optimized Rectilinear Regions for Association Rules, Proc. of the Third Conference on Knowledge Discovery and Data Mining (KDD'97), pp. 96-103, Los Angels, August 1997.

[18] M. J. Zaki, Efficient Enumeration of Frequent Sequences, 7th International Conference on Information and Knowledge Management, pp. 68-75, Washington DC, November 1998.

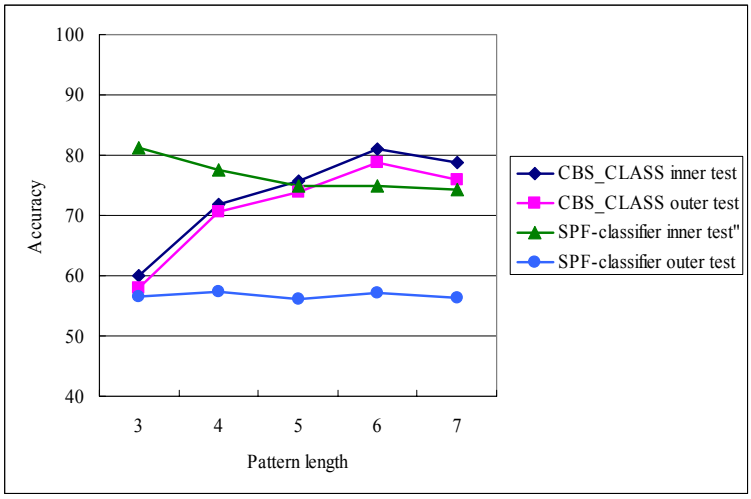

Figure 8. Comparative results by varying *pattern_len*.

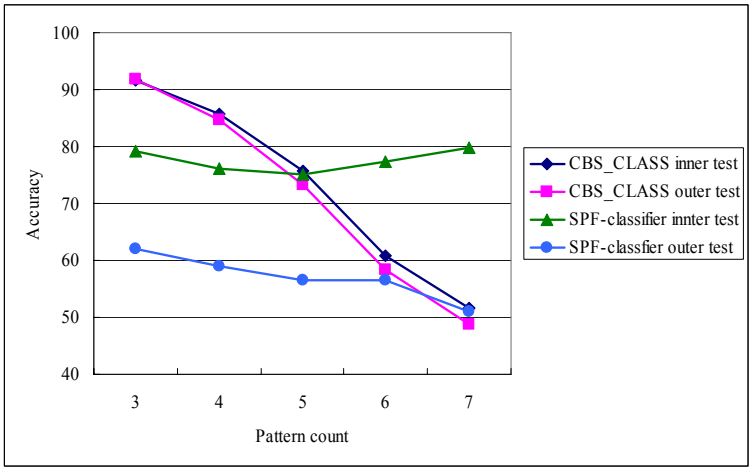

Figure 9. Comparative results by varying *pattern_count*.

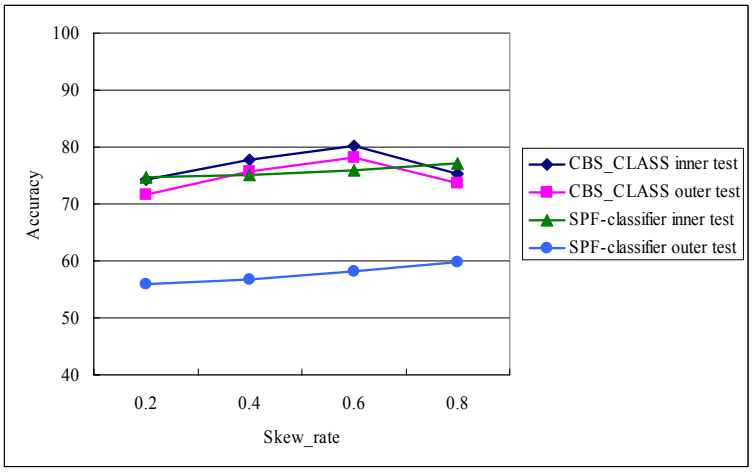

Figure 10. Comparative results by varying *data_skew*