# Efficiently Finding the Best Parameter for the Emerging Pattern-Based Classifier PCL[⋆]

Thanh-Son Ngo[1], Mengling Feng[2] Guimei Liu[1], and Limsoon Wong[1]

[1] National University of Singapore,
{ngothanh,liugm,wongls}@comp.nus.edu.sg,
[2] Institute for Infocomm Research,
mfeng@i2r.a-star.edu.sg

**Abstract.** Emerging patterns are itemsets whose frequencies change sharply from one class to the other. PCL is an example of efficient classification algorithms that leverage the prediction power of emerging patterns. It first selects the top-K emerging patterns of each class that match a testing instance, and then uses these selected patterns to decide the class label of the testing instance. We study the impact of the parameter K on the accuracy of PCL. We have observed that in many cases, the value of K is critical to the performance of PCL. This motivates us to develop an algorithm to find the best value of K for PCL. Our results show that finding the best K can improve the accuracy of PCL greatly, and employing incremental frequent itemset maintenance techniques reduces the running time of our algorithm significantly.

## 1 Introduction

Classification is the task of learning from one data set and making predictions in another data set. The learning data is called training data which consists of entities with their labels. The other data set is called testing data which consists of entities without labels, and the classifying task is to make prediction of their labels based on what we learn from the training data. Many classifiers have been proposed in the literature. Here, we focus on pattern-based classifiers.

Frequent patterns are patterns that appear frequently in the dataset. Frequent patterns whose supports in the training data set change significantly from one class to the other are used to construct the classifiers. These patterns are called emerging patterns (EP) [9, 10, 16] and these patterns are applied to testing instances to predict their class memberships. As a classifier may generate many rules for the same class, aggregating their discrimination power gives better prediction results [4, 8, 10, 20]. Using EPs has the advantage that they not only predict the class labels but also provide explanation for the decision.

Jumping emerging patterns (JEP) are a special type of EPs that have non-zero occurrence in one class and zero occurrence in all other classes [10, 11]. PCL [9, 10] is a classifier based on aggregating the prediction power of frequent JEPs. Given a test instance $t$, PCL selects the top-K JEPs with the highest

⋆ Supported by A*STAR grant (SERC 072 101 0016)

supports that are contained in $t$ from each class and then computes the score for each class using these selected patterns. PCL assigns the class label with the highest score to the testing instance. The choice of a good value for K is tricky and its optimal value varies on different datasets, as shown in Fig. 1. Remarkably, the problem of choosing the best value of K has not been investigated previously.

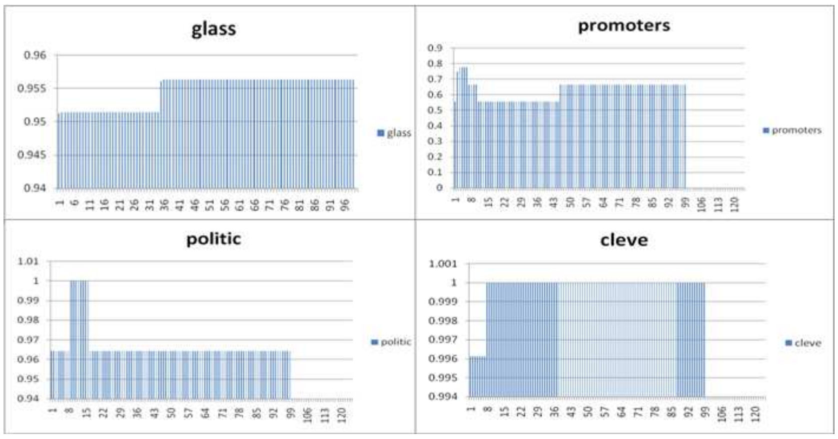


**Fig. 1.** Accuracy of PCL on four datasets with different values of K. X-axis shows different values of K and Y-axis shows the accuracy. When K increases, the accuracy does not always increase.

Here, to avoid overfitting K to the data, we sample many subsets of the training data to get the value of K that appears the best on average. By this method, we maximize the likelihood that the chosen K will produce the best results in the whole dataset. Our main contributions are summarized as follows: (i) We revisit the PCL algorithm [9, 10] and propose a method to find the most appropriate parameters to improve the performance of PCL. (ii) We introduce a method to speed up the proposed algorithm as well as cross-validation methodology for frequent-pattern-based classification algorithms in general.

## 2 Related work

Emerging patterns are a special type of frequent patterns. They occur frequently in one class but rarely in other classes. Emerging pattern-based classification benefits enormously from the advancement of frequent pattern mining algorithms and better understanding of the patterns space. Many methods have been proposed to select a good set of patterns for constructing classifiers [15]. Most of these algorithms first generate frequent patterns satisfying a certain minimum support constraint, and then select patterns based on some criteria and make predictions based on the selected patterns. These algorithms mainly differ in how patterns are selected and how class labels are determined.

The first class of methods pick the best pattern to classify testing instances, like CBA [14] and MMAC [18]. CBA first derives classification rules from frequent patterns, and then ranks the rules in descending order of their confidence. A very small set of rules is then selected to maximize the classification accuracy on the training data. The class label of a new instance is decided by the first rule that matches the instance. MMAC extends CBA to multiple classes and labels. Only a small set of patterns are selected for classification. So it is possible that a new testing instance does not contain any EP that is selected.

The second class of methods treats frequent patterns as additional features and then use classical algorithms to classify the data [2, 3]. Cheng et al. [2, 3] build a connection between pattern frequency and discriminative measures such as information gain score, and develop a strategy to set minimum support in frequent pattern mining for generating useful patterns for classification. Based on this strategy, coupled with a proposed feature selection algorithm, selected EPs are used as additional features to build high quality classifiers. The results show that using EPs as additional features can improve the accuracy of C4.5 and SVM. The main drawback of this approach is that the intuitiveness of the pattern-based classifiers may be lost.

The last class of methods selects the top-K patterns and aggregates the predictive power of the selected patterns. They include CAEP [4], iCAEP [20], PCL [9, 10], CEP [1], CPAR [17], CMAR [8] and HARMONY [19]. CAEP uses EPs with high growth rate to do classification. iCAEP aggregates the prediction power of EPs based on information theory. PCL uses only JEPs. JEPs occur in one and only one class, which may be too restrictive in some cases. CEP relaxes this constraint by putting an upper bound on the number of occurrences of the EPs in other classes. CPAR uses the expected accuracy to select classification rules. It compares the average expected accuracy of the best K rules of each class and chooses the class with the highest expected accuracy as the predicted class. CMAR uses a weighted measure to select rules, and the score of a class is also calculated using this weighted measure. HARMONY directly mines the final set of classification rules by using an instance-centric rule generation approach.

The aggregation-based algorithms generally show better performance than other algorithms. However, the value of K is critical to their performance in many cases. Here, we use PCL as an example to study how to select proper values for parameter K to maximize the accuracy of the aggregation-based algorithms.

## 3 Preliminaries

Let $I = \{i_1, i_2, .., i_n\}$ be a set of distinct literals called items. An itemset or a pattern is a subset of $I$. A transaction is a non-empty set of items. Let $C = \{C_1, C_2, .., C_k\}$ be a set of distinct labels called class labels. A transactional database consists of a set of transactions associated with their labels. A classification task involves two phases: training and testing. In training, the class labels are revealed and in testing, class labels are hidden. The classifier builds a model based on the training set and uses this model to predict the class labels of transactions in the testing set. The accuracy of a classifier $A$ is the proportion of test-instances which are correctly classified by $A$.

A pattern $P$ covers a transaction $t$ if $P \subseteq t$. The support of $P$ is the number of transactions that are covered by $P$. We use $sup(P, D)$ to indicate the support of $P$ in the dataset $D$. We use $P|D$ to indicate the set of transactions in $D$ that contain $P$. Given a 2-class dataset, jumping emerging patterns (JEP) are patterns whose frequency in one class is non-zero and in other class is zero. An $EP_i$ is called a JEP from class $A$ to class $B$ if its support in class $A$ is zero. A pattern $P$ is a generator if and only if for every $P' \subset P$, $sup(P', D) > sup(P, D)$. A JEP generator is both a generator and a JEP.

## 4 Algorithms

### 4.1 PCL

We present here an overview of PCL [9, 10]. The dataset $D$ is divided into positive and negative classes. Given a test-instance $t$, two sets of JEPs that cover $t$ are used: $EP^+_{t1}, EP^+_{t2}, .., EP^+_{tn}$ from negative to positive classes and $EP^-_{t1}, EP^-_{t2}, .., EP^-_{tn}$ from positive to negative classes. The K most frequent JEPs that cover $t$ are sorted in descending order of their supports. Suppose the set of JEPs (based on the training set) from negative to positive classes are $EP^+_1, EP^+_2, .., EP^+_n$ in descending order of their supports. Similarly, $EP^-_1, EP^-_2, .., EP^-_n$ is the set of JEPs from positive to negative. It is hypothesized that if $t$ belongs to one class, it should contain more JEPs of this class than the other class. So Li and Wong [9] formulated the scores of $t$ with respect to the two classes as below.

$$Score(t, +) = \frac{\sum_{i=1}^{k} sup(EP^+_{ti}, D)}{\sum_{i=1}^{k} sup(EP^+_i, D)} \qquad Score(t, -) = \frac{\sum_{i=1}^{k} sup(EP^-_{ti}, D)}{\sum_{i=1}^{k} sup(EP^-_i, D)}$$

### 4.2 PSM

Maintenance of EPs was first introduced in [11], though a complete solution for insertion and deletion was not discussed. Here, we describe a more efficient method called PSM for complete maintenance introduced recently in [5].

PSM is a maintenance algorithm for frequent pattern space. It is based on the GE-tree, an effective data structure described in [6, 7] for enumerating generators. Frequent generators are stored in the GE-tree and the tree is incrementally adjusted when new transactions are added or existing transactions are removed. Each node in the tree represents a generator. To find EPs, we modify the GE-tree so that each node stores both positive and negative supports of the corresponding generator. In addition to frequent generators, GE-tree maintains a negative border which comprises infrequent generators whose immediate subsets are all frequent [5, 6]. The negative border helps generate new frequent generators and equivalence classes efficiently when transactions are added or removed.

Computating frequent generators is expensive. The benefit of PSM is that a new set of frequent generators does not need to be computed from scratch when the data is modified. As a small change in dataset seldom causes a big change in the set of frequent patterns, PSM is effective for pattern space maintenance.

### 4.3 rPCL and ePCL

A good choice of K has a big impact on prediction results in PCL. We propose a method to tackle this problem as follows. According to the Central Limit Theorem, the distribution of accuracy will behave like normal distribution. Indeed, Fig. 2 suggests the convergence of average classification accuracy in training data to the real value of accuracy in the whole dataset. So we simulate the actual process of classification in training set and choose the value of K that maximizes the mean accuracy. The simulation is run repeatedly to determine which value of K appears as the best on average. By the Central Limit Theorem, the average accuracy of each K approaches the true accuracy of that value of K, given sufficient number of simulation runs. Thus, the value of K that has the best mean accuracy in the simulation runs will also perform well in the whole set of data.

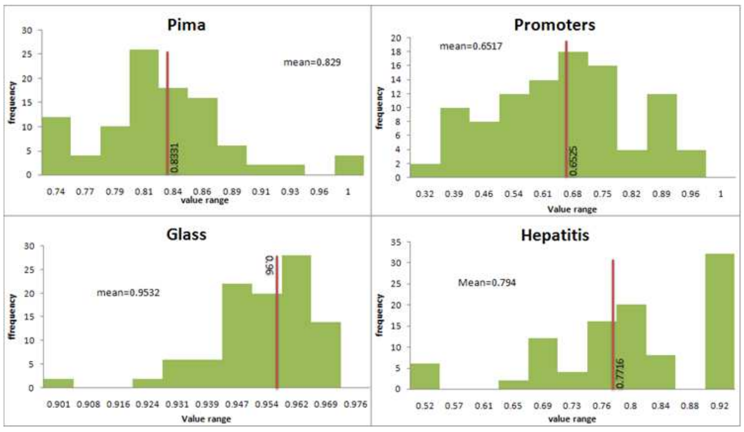
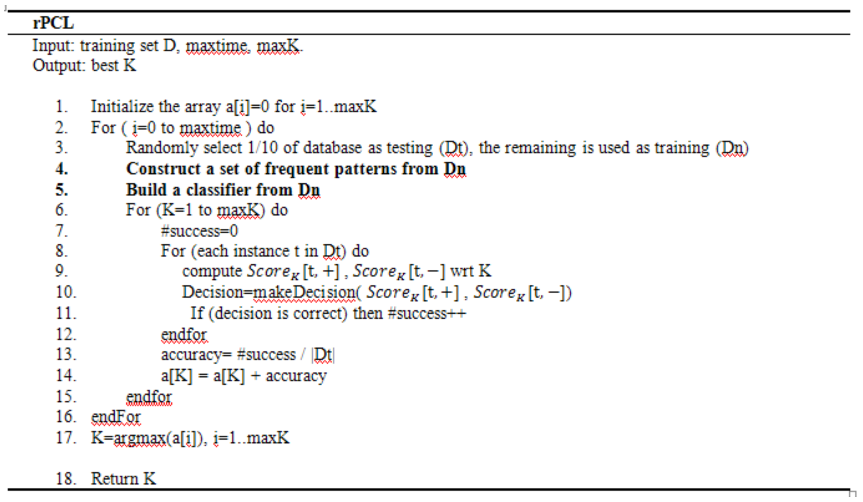


**Fig. 2.** Distribution of accuracies across 100 runs given a fixed K=10 over training set. The vertical red line indicates the actual value of accuracy when we evaluate the same K in the whole dataset. The means are shown to be close to the actual accuracy.

We describe two algorithms rPCL and ePCL. Algorithm rPCL is a direct but naive solution for the method above. We use it as a benchmark to understand where inefficiencies might be and to assess the improvement of a more efficient method using maintenance. ePCL is the fast version where PSM is used. It produces the same results as rPCL but runs many times faster.

In rPCL (Fig. 3) we use a technique called repeated random sub-sampling validation as a framework to assess the best parameter for PCL. It involves several rounds, in each round the training set is randomly divided into new complementary testing and training set. For each K, we perform this procedure to determine the expected accuracy of the classifier wrt this K. The chosen K is the one which returns the best average accuracy over these runs. 10-fold cross validation is popular for accessing classifier performance. So, we subsample 10% of training set as testing subsample and 90% as training subsample.

In rPCL, makeDecision simply decides if the classifier with $Score_K[t, +]$ and $Score_K[t, -]$ correctly predicts the class of $t$, wrt to the current K. At step 9, we can compute $Score_K[t, +]$ from $Score_(K - 1)[t, +]$ in constant time. To achieve this, we use a vector to store $Score_K[t, +]$ and $Score_K[t, -]$ for all $t \in Dt$. To determine a good value of K, a significantly large number of iterations maxtime is performed and all values of K in the range 1..maxK are considered.



```
rPCL
Input: training set D, maxtime, maxK.
Output: best K

1.  Initialize the array a[i]=0 for i=1..maxK
2.  For ( i=0 to maxtime ) do
3.      Randomly select 1/10 of database as testing (Dt), the remaining is used as training (Dn)
4.      Construct a set of frequent patterns from Dn
5.      Build a classifier from Dn
6.      For (K=1 to maxK) do
7.          #success=0
8.          For (each instance t in Dt) do
9.              compute $Score_K$[t, +] , $Score_K$[t, −] wrt K
10.             Decision=makeDecision( $Score_K$[t, +] , $Score_K$[t, −])
11.               If (decision is correct) then #success++
12.         endfor
13.         accuracy= #success / |Dt|
14.         a[K] = a[K] + accuracy
15.     endfor
16. endFor
17. K=argmax(a[i]), i=1..maxK

18. Return K
```

**Fig. 3.** The rPCL algorithm.

Constructing a set of frequent EPs from the training set (step 4) is computationally very expensive. It involves frequent pattern mining. And this step is repeated many times with different Dt and Dn. Because the testing-training separations are largely similar among all the runs, we should not need to construct the set of EPs from scratch in such a naive fashion. PSM [5, 6] is used to achieve this purpose. PSM allows us to adjust the set of frequent patterns when the training fold is changed. We only need to construct the frequent patterns space at the beginning and repeatedly use PSM to maintain the set of rules when the sub-sampling folds are changed. That is, we only need to mine frequent patterns once at the beginning. With this innovation, we present an improved algorithm called ePCL (Fig. 4) which stands for enhanced PCL.

The set of frequent generators are incrementally maintained. Line 5 and 17 show the execution of PSM: PSM.delete is invoked when a fold is removed from the training set and PSM.add is invoked to add this fold into original set. That eliminates building a new set of frequent patterns when we run the simulation with a new testing-training separation. Fig. 5 is the workflow of rPCL and ePCL.

## 4.4 Complexity analysis

We compare the theoretical improvement of ePCL from rPCL. In rPCL, the outer loop repeats maxtime, which is the number of runs needed to find the best K. In each loop, we need to run PCL classifier which involves a frequent pattern mining step ($FPM$) and evaluation for all K from 1 to maxK. The

```
ePCL
Input: training set D, maxtime, maxK.
Output: best K

1.  Construct a frequent patterns maintenance FG which consists of frequent generators from D
2.  Initialize the array a[i]=0 for i=1..maxK
3.  For ( i=0 to maxtime ) do
4.      Randomly select 1/10 of database as testing (Dt), the remaining is used as training (Dn)
5.      PSM.delete( Dt )
6.      Build a classifier from Dn
7.      For (K=1  to maxK ) do
8.          #success=0
9.          For (each instance t in Dt) do
10.             compute $Score_K[t,+]$ , $Score_K[t,-]$ wrt K
11.             Decision=makeDecision( $Score_K[t,+]$ , $Score_K[t,-]$)
12.              If (decision is correct) then #success++
13.         endfor
14.         accuracy= #success / |Dt|
15.         a[K] = a[K] + accuracy
16.     endfor
17.     PSM.add( Dt )
18. endfor
19. K=argmax(a[i]), i=1..maxK

20. Return K
```

**Fig. 4.** ePCL algorithms. The highlighted codes indicate the involvement of PSM.

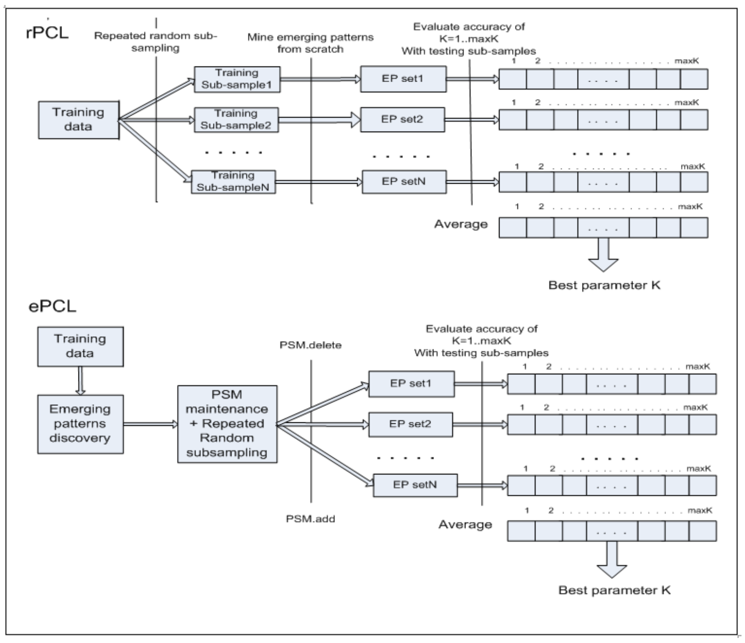

rPCL
Repeated random sub-sampling
Mine emerging patterns from scratch
Evaluate accuracy of K=1..maxK With testing sub-samples
Training data
Training Sub-sample1
Training Sub-sample2
Training Sub-sampleN
EP set1
EP set2
EP setN
1
2
maxK
Average
Best parameter K
ePCL
Training data
Emerging patterns discovery
PSM maintenance + Repeated Random subsampling
PSM.delete
PSM.add
Evaluate accuracy of K=1..maxK With testing sub-samples
EP set1
EP set2
EP setN
Average
Best parameter K

**Fig. 5.** Workflow for rPCL and ePCL. EP set indicates the set of EPs used to construct PCL.

total time is $maxtime * (FPM + PCL(maxK))$. $PCL(maxK)$ is the time to compute scores for $K = 1..maxK$. This step evolves visiting entire frequent pattern set to get the top K patterns and compute scores accordingly for each test instance. In ePCL, we only need to build frequent patterns once and incrementally adjust the pattern space. The total time is $FPM + maxtime * (|D|/10 * maintenance + PCL(maxK))$, where maintenance is the running time for maintaining one transaction, $|D|$ is the size of the dataset. According to [5], PSM is much more efficient than mining-from-scratch algorithms. At 10% of the data size, the speed-up is at least 3 times faster for incremental and 5 times faster for decremental maintenance than mining from scratch.

## 5 Experimental studies

### 5.1 Experiment setup

Experiments are conducted using 18 data sets of various sizes from UCI repository. Continuous attributes are discretized by the entropy method. Table 1 gives details of these data sets. We evaluate the performance on two-class datasets. For multi-class datasets, we select one class as positive and the remaining as negative. The process is done for all classes. PCL can be extended to handle multiple-class datasets; however, the method of choosing parameter is still applicable. 10-fold cross validation is used to assess the efficiency and average accuracies are reported. The datasets are separated in such a way that the proportion of each class is consistent in both testing and training.

| Dataset | # attributes | # instances | # continuous attributes | # nomial attributes |
|---|---|---|---|---|
| mushroom | 4 | 8124 | 4 | 0 |
| iris | 8 | 150 | 8 | 0 |
| Glass | 22 | 214 | 0 | 22 |
| zoo | 16 | 101 | 0 | 16 |
| Promoter | 18 | 106 | 0 | 18 |
| Vote | 10 | 435 | 10 | 0 |
| Splice | 19 | 3190 | 0 | 19 |
| Hepatitis | 59 | 155 | 0 | 59 |
| Pima | 61 | 768 | 0 | 61 |
| Hypothyroid | 25 | 3163 | 7 | 18 |
| Breast | 16 | 3190 | 0 | 16 |
| Cleve | 10 | 2727 | 0 | 10 |
| German | 12 | 2700 | 0 | 12 |
| Lymph | 19 | 1332 | 0 | 19 |
| Vehicle | 19 | 3809 | 0 | 19 |
| Waveform | 20 | 9000 | 0 | 20 |
| Wine | 14 | 178 | 0 | 14 |
| Tic-tac-toe | 10 | 4312 | 0 | 10 |

**Table 1.** Datasets information.

For the original PCL, we use the frequent pattern mining algorithm provided by [13]. The experiments are done in Windows machine with 2.6 GHz processor and 1G memory. We assess the accuracy improvement of rPCL over PCL and running time comparison of ePCL and rPCL. Time is measured in seconds.

When we compute the score of an instance in two classes, it is possible that the scores are both zero. Such a test-instance is not covered by any pattern and therefore unclassifiable by both PCL and ePCL. The average percentage of unclassified data is 11% and there is no dataset with more than 20%. We do not include these cases in our accuracy computation. The purpose is to evaluate the improvement of rPCL over the original PCL only.

### 5.2 Parameters setting

All the patterns used in PCL and its improved versions are JEP generators [10]. The exception is an early paper [8]; it uses the "boundary" JEPs, which are defined in [9] as those JEPs where none of their proper subsets are JEPs. Here, we use JEP generators as discriminative patterns because:

– It is known that the JEP space is partitioned into disjoint equivalence classes [9, 10]. So the set of JEP generators (which correspond to the most general patterns in each equivalence class) adequately characterizes the data. In contrast, boundary JEPs are insufficient to capture all equivalence classes.
– The minimum description length principle is a well-accepted general solution for model selection problems. The choice of JEP generators is in accord with this principle [12]. Also, our experiments show that JEP generators are less noise-sensitive than boundary JEPs.
– JEP generators are efficient to compute. We can make use of many efficient frequent patterns mining algorithms [12].

For rPCL and ePCL, we set maxtime to 50 and we run K in a range from 1 to 50. We limit the range to 50 because small-frequency patterns have minimal prediction power over the top ones. Patterns ranked higher than 50 generally have low support. As shown in Fig. 1, the predictions stabilize after $K = 50$, so no need to consider $K > 50$. For original PCL, we use $K = 10$ as suggested in [9]. Minimum support is set to make sure enough EPs are found.

### 5.3 Efficiency

**PCL with generators and boundary EPs.** For PCL, we have choices over which type of frequent patterns are used to make predictions. We have done experiments to justify our choice of JEP generators, as suggested in [9], rather than boundary JEPs. We want to test the robustness of PCL in the case of noise with these two types of patterns. When a part of the original dataset is missing, the noisy dataset reduces the accuracy of the classifier. Fig. 6 shows accuracy of PCL with generators (gen PCL) and boundary JEPs (boundary PCL) for different levels of missing rate (We assume items in the dataset are missing with certain rate): 0%, 10%, 20%, 30%, 40% and 50%. The accuracies are average over 18 datasets. Gen PCL is less noise-sensitive than boundary PCL.

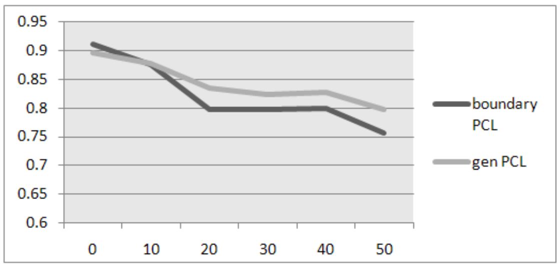


**Fig. 6.** Accuracy of gen PCL and boundary PCL in the presence of missing information.

**PCL and ePCL.** We compare the accuracy of the original PCL and ePCL. Since rPCL and ePCL give the same results, we do not show the accuracy of rPCL here. Table 2 shows accuracy of ePCL and the original PCL in 18 datasets. Overall, the improvement is 3.19%. In some datasets like Promoters, Wine and Hepatitis, we get improvement of 26%, 11% and 15% respectively.

| Datasets | ePCL | PCL | Improve (%) | Datasets | ePCL | PCL | Improve (%) |
|---|---|---|---|---|---|---|---|
| Mushroom | 1 | 1 | 0 | Iris | 0.9999 | 0.9658 | 3.534591 |
| Glass | 0.9714 | 0.9675 | 0.404967 | Zoo | 0.9997 | 0.9528 | 4.925911 |
| Promoter | 0.8216 | 0.6525 | 25.92549 | Vote | 0.9439 | 0.9492 | -0.55892 |
| Splice | 0.6667 | 0.6667 | 0 | Hepatitis | 0.8872 | 0.7716 | 14.98068 |
| Pima | 0.8192 | 0.8331 | -1.67043 | Hypothyroid | 0.9962 | 0.9861 | 1.017527 |
| Breast | 0.9912 | 0.9907 | 0.050572 | Cleve | 0.9644 | 0.9959 | -3.16567 |
| German | 0.9644 | 0.994 | -2.97462 | Lymph | 1 | 1 | 0 |
| Vehicle | 0.9569 | 0.9437 | 1.390222 | Waveform | 0.8347 | 0.8053 | 3.643507 |
| Wine | 0.97892 | 0.8789 | 11.3941 | Tic-tac-toe | 0.9512 | 0.965 | -1.42591 |

**Table 2.** Accuracy comparison of ePCL and PCL.

Recall in PCL, to classify one test instance, one score is computed for each class. Some instances received zero score in one class and non-zero score in the other class; thus the association rules vote unanimously to one class. We call these the easy cases and the rest are difficult cases. The value of K has a much greater impact on the difficult cases. Table 3 shows the accuracy of original PCL and ePCL in difficult cases. Although we do not show statistics for easy cases here, the accuracy is close to 100% for most easy cases. We do not show datasets with too few difficult cases since there are not enough samples to evaluate accurately.

**Performance.** We compare the running time of rPCL, ePCL and original PCL. The performance of ePCL compared to rPCL is good, demonstrating the superiority of using pattern maintenance. Table 4 shows that ePCL is an order of magnitude faster in many cases. In one case (vehicle.dat), rPCL could not complete after more than one day, but ePCL completed within 4 hours. Nevertheless, ePCL takes more time than PCL due to the repeating part; fortunately, this is compensated by the much better accuracy of ePCL.

| Datasets | Difficult cases (%) | PCL | ePCL | Datasets | Difficult cases (%) | PCL | ePCL |
|---|---|---|---|---|---|---|---|
| mushroom | 0 | - | - | Iris | 0 | - | - |
| Glass | 33.94 | 0.9071 | 0.9183 | zoo | 26.53 | 0.8274 | 0.9989 |
| Promoter | 49.02 | 0.3123 | 0.647 | Vote | 16.31 | 0.6978 | 0.6663 |
| Splice | 100 | 0.6667 | 0.6667 | Hepatitis | 100 | 0.7716 | 0.8872 |
| Pima | 12.10 | 0 | 0 | Hypothyroid | 20 | 0.9327 | 0.9813 |
| Breast | 10.90 | 0.917 | 0.9214 | Cleve | ¡1 | - | - |
| German | 10.60 | 0.9449 | 0.6745 | Lymph | 0 | - | - |
| Vehicle | 3.7453 | 0 | 0 | Waveform | ¡2 | - | - |
| Wine | 31 | 0.8788 | 0.9789 | Tic-tac-toe | 30.2 | 0.553 | 0.578 |

**Table 3.** Accuracy comparison of ePCL and PCL in difficult cases.

| Datasets | PCL | rPCL | ePCL | Speed up (rPCL/ ePCL) | Datasets | PCL | rPCL | ePCL | Speed up (rPCL/ ePCL) |
|---|---|---|---|---|---|---|---|---|---|
| mushroom | 2.75 | 84 | 35 | 2.4 | Iris | 2 | 99 | 3 | 33 |
| glass | 8.5 | 120 | 2 | 60 | zoo | 5 | 291 | 7 | 41.5 |
| promoters | 1 | 51 | 7 | 7.2 | Vote | 0.75 | 47 | 11 | 4.2 |
| splice | 2.5 | 129 | 4 | 32.2 | hepatitis | 0.5 | 38 | 3 | 12.6 |
| pima | 0.75 | 42 | 9 | 4.6 | hypothyroid | 2 | 105 | 43 | 2.4 |
| breast | 2.5 | 109 | 60 | 1.8 | cleve | 4.25 | 181 | 123 | 1.4 |
| german | 11 | 513 | 611 | 0.8 | lymph | 27.5 | 1224 | 512 | 2.3 |
| vehicle | 2518 | too long | 6966 | A lot | waveform | 117 | 5738 | 2507 | 2.3 |
| wine | 3.25 | 188 | 17 | 11.1 | tictactoe | 13 | 88 | 43 | 2 |

**Table 4.** Performance comparison.

## 6 Discussion and conclusion

Cross-validation is a technique to assess the performance of classification algorithms. Typically the testing set is much smaller than the training set and a classifier is built from the training set which is largely similar between runs. Our framework of maintenance can be applied to efficiently perform cross-validation for pattern-based classification. Recall PSM allows us to maintain the set of frequent generators while adding or deleting a relatively small portion of the dataset. We only need to maintain a single frequent pattern set and incrementally adjust using PSM. In addition, for more efficient computation, PSM can be extended to handle multiple additions/deletions at the same time by organizing transactions in a prefix tree.

We showed the importance of finding the top K patterns for PCL. We introduced a method to perform the task efficiently, making use of the PSM maintenance algorithm [5]. The maintenance framework can be applied to general cross-validation tasks which involve frequent update of the pattern space. Our experiments showed improvement in accuracy. However, we compromised on complexity. We hope to get a better implementation for ePCL, thus minimizing the running time to close to original PCL and implement the mentioned extensions.

## References

1. J. Bailey, T. Manoukian, K. Ramamohanarao. Classification using constrained emerging patterns. *Proc 4th WAIM*, pp. 226–237, 2003.
2. H, Cheng, et al. Discriminative frequent pattern analysis for effective classification. *Proc 23th ICDE*, pp. 716–725, 2007.
3. H. Cheng, et al. Direct discriminative pattern mining for effective classification. *Proc 24th ICDE*, pp. 169–178, 2008.
4. G. Dong, et al. CAEP: Classification by Aggregating Emerging Patterns. *Proc 2nd Intl Conf on Discovery Science*, pp. 30–42, 1999.
5. M. Feng. *Frequent Pattern Maintenance: Theories and Algorithms.* PhD thesis, Nanyang Technological University, 2009.
6. M. Feng, et al. Negative generator border for effective pattern maintenance. *Proc 4th Intl Conf on Advanced Data Mining and Applications*, pp. 217–228, 2008.
7. M. Feng, et al. Evolution and maintenance of frequent pattern space when transactions are removed. *Proc 11th PAKDD*, pp. 489–497, 2007.
8. W. Li, J. Han, J. Pei. CMAR: Accurate and efficient classification based on multiple class-association rules. *Proc 1st ICDM*, pp. 369–376, 2001.
9. J. Li, L. Wong. Solving the fragmentation problem of decision trees by discovering boundary emerging patterns. *Proc 2nd ICDM*, pp. 653–656, 2002.
10. J. Li, L. Wong. Structural geography of the space of emerging patterns. *Intelligent Data Analysis*, 9(6):567–588, 2005.
11. J. Li, K. Ramamohanarao, G. Dong. The space of jumping emerging patterns and its incremental maintenance algorithms. *Proc of 17th ICML*, pp. 551–558, 2000.
12. J. Li, et al. Minimum description length principle: Generators are preferable to closed patterns. *Proc 21st Natl Conf on Artificial Intelligence*, pp. 409–415, 2006.
13. H. Li, et al. Relative risk and odds ratio: A data mining perspective. *Proc 24th PODS*, pp. 368–377, 2005.
14. B. Liu, W. Hsu, Y. Ma. Integrating classification and association rule mining. *Proc 4th KDD*, pp. 80–86, 1998.
15. K. Ramamohanarao, H. Fan. Patterns based classifiers. *Proc 16th WWW*, pp. 71–83, 2007.
16. K. Ramamohanarao, J. Bailey. Discovery of emerging patterns and their use in classification. *Proc 16th AI*, pp. 1–12, 2003.
17. F. A. Thabtah, P. Cowling, Y. Peng. MMAC: A new Multi-class Multi-label Associative Classification approach. *Proc 4th ICDM*, pp. 217–224, 2004.
18. X. Yin, J. Han. CPAR: Classification based on Predictive Association Rules. *Proc 3rd SDM*, pp. 331–335, 2003.
19. J. Wang, G. Karypis. HARMONY: Efficiently mining the best rules for classification. *Proc 5th SDM*, pp. 205–216, 2005.
20. X. Zhang, G. Dong, K. Ramamohanarao. Information-based classification by aggregating emerging patterns. *Proc 2nd Intl Conf on Intelligent Data Engineering and Automated Learning*, pp. 48–53, 2000.