PatHT: An Efficient Method of Classification over Evolving Data Streams

Meng Han, Jian Ding, and Juan Li
School of Computer Science and Engineering, North Minzu University, China

Abstract: *Some existing classifications need frequent update to adapt to the change of concept in data streams. To solve this problem, an adaptive method Pattern-based Hoeffding Tree (PatHT) is proposed to process evolving data streams. A key technology of a training classification decision tree is to improve the efficiency of choosing an optimal splitting attribute. Therefore, frequent patterns are used. Algorithm PatHT discovers constraint-based closed frequent patterns incremental updated. It builds an adaptive and incremental updated tree based on the frequent pattern set. It uses sliding window to avoid concept drift in mining patterns and uses concept drift detector to deal with concept change problem in procedure of training examples. We tested the performance of PatHT against some known algorithms using real data streams and synthetic data streams with different widths of concept change. Our approach outperforms traditional classification models and it is proved by the experimental results.*

Key words: *Data mining; decision tree; data stream classification; closed pattern mining; concept drift.*

Received November 13, 2015; accepted April 12, 2018

1. Introduction

The explosion in the variety, volume and velocity of data is generated by the increasing availability of phones, internet, and sensors. This data is often referred as data streams [1]. Data stream classification is the way that knowledge and information from continuous data is extracted [12]. In classification, evolving data streams classification is one of the most complex problems [17]. The first problem is concept drifts. The distribution of the data streams is not stable, and it varies over time. Other problems include large number of examples and limited time or memory requirements [16, 24].

Among classifier technique, decision tree is a very prevalent because its advantages are easy to interpret and visualize the tree models [4]. Hoeffding bound is a common used estimating split criterion. Such as algorithms Very Fast Decision Tree (VFDT) [10], Concept adapting Very Fast Decision Tree (CVFDT) [20], VFDTc [14], VFT [21] are based on Hoeffding bound. Algorithm VFDT is the earlier classification methods to process data streams. The shortage of VFDT is that it can't deal with concept drift problem. Algorithm CVFDT has two main differences from VFDT. One is to handle concept drift problem, the other is to handle examples in a sliding window model [4]. Two algorithms VFDTc and UFFT are based on Hoeffding Tree which are used to deal with concept drift problem and numeric attributes over data streams [4]. Algorithms HAT [21], HOT [2], AdoHOT [27] and ASHT [3] are also Hoeffding trees. These algorithms use ADaptive WINdowing (ADWIN) [3] to deal with concept drift. Algorithm streamDM [6] is an ensemble method that uses adaptive decision trees efficiently and easily. It also uses Hoeffding bound.

Recent years, researchers propose a variety of methods for discovering frequent patterns over data streams. Algorithms FIS_EDS [11], SysTree [7] and CanTree-GTree [22] are used to discover frequent patterns which meet error bound and minimum support. These methods don't distinguish recent and historical examples. Time Decay Model (TDM) is used in algorithms Sliding Window Pattern tree (SWP-Tree) [8] and TwMinSwap [23] to set different weights of recent and historical examples. They emphasize the importance of new example and can get more reasonable pattern sets. However, complete pattern sets are mined out and a large number of useless patterns are contained. In order to reduce a quantity of patterns, compressed patterns should be discovered. Closed patterns are lossless compared with other compressed patterns. Algorithms Moment [9], TDMCS*[18], FLMCFI [26], CloStream [29] and CloStream*[30] use sliding window to discover closed patterns on data streams.

Although many classification methods have been proposed, studies find that frequent patterns can be used to build high quality classification models. The advantages of pattern-based classification methods lie in:

1. Un-frequent itemsets may be caused by random noises which are harmful to classification methods. But frequent patterns always carry reliable information gains to construct methods.
2. Generally, patterns have more information gains than a single attribute.
3. The discovered patterns are always simple and easy to explain. Therefore, interesting, frequent and

distinguished pattern can be used for effective classification and may lead to high accuracy.

In this paper, we focus on mining closed frequent patterns on data streams, and study classification decision trees based on these patterns. Our contributions can be summarized as follows:

1. Existed data stream classification cycle includes three steps: input-learning-model [27]. In order to improve the efficiency of training model and improve classification accuracy, we propose the four steps of classification cycle IPLmodel: Input-Pattern-Learning- model.
2. Propose an algorithm to discover closed frequent patterns incremental updated based on IPLmodel. All patterns must contain attribute and class label.
3. The sliding window model is used to discover frequent patterns on recent examples. We use top-*k* frequent patterns to build decision trees in order to improve the efficiency of choosing optimal splitting attributes. We use ADWIN [3] to detect concept change in evolving data streams.

2. Frequent Patterns

A data stream $DS = \{Ex_1, Ex_2, \ldots, Ex_m, \ldots, Ex_{new}\}$ is an orderly, continuous, unrestricted flow of examples instances, or transactions. An example Ex_m (m=1, 2, …) generated at a time step m, is a set of $<X_m, C_m>$ pairs. In which X_m is an n-dimensional feature vector that consists of n attribute values and C_m is a class label.

In some time-sensitive data streams applications, users have most interest in recent examples. Common methods for such cases are used Sliding Window Model (SWM). The latest W examples in a data stream DS are contained in a sliding window of size W. Function $freq(Q)$ means the frequency of pattern Q, which is the number of examples including itemsets Q. Function $support(Q)$ means the support of Q in a window, which is defined as $freq(Q)/W$. In this paper, we discover closed frequent patterns(CFPs) over data stream on the basis of the SWM. The discovered CFPs meet class-constraints. The definitions of frequent patterns (FPs) and CFPs are shown in Definitions 1 and 2, the class-constraints is shown as follow.

- Definition 1 (FPs) Variable θ ($\theta \in [0, 1]$) is a minimum support threshold. Itemset Q is a frequent pattern in SW if $support(Q) \geq \theta$.
- Definition 2 (CFPs) Itemset Q is a closed frequent pattern in SW, if Q is a frequent itemset in SW and there is not its parent itemset Z in SW such that $support(Z) = support(Q)$.
- *Class-constraints*: A pattern Q must satisfy two constraints. (1) The form is like $<X, C>$, which must contain at least one attribute value and one class value. (2) It is closed.
- *Example* 1. There are 8 examples in Table 1. Length of each example is 5, including 4 attributes and 1 class. The distinguishable values of attributes $\{A_1, A_2, A_3, A_4\}$ are {3, 3, 2, 2} separately. There are 2 distinguishable values {yes, no} of class. Variable A_i is the ith attribute. Variable A_{ij} is the jth value of A_i. Variable C_k is the kth value of class label. Variable V_{ijk} denote the number of A_{ij} under the condition of C_k.

Table 1. Data stream.

Example	A_1	A_2	A_3	A_4	Class
Ex_1	a+	b+	c+	d+	yes
Ex_2	a+	b+	c+	d-	yes
Ex_3	a+	b-	c+	d+	yes
Ex_4	a+	b-	c-	d-	no
Ex_5	a+	b	c+	d+	yes
Ex_6	a+	b	c+	d-	no
Ex_7	a-	b-	c-	d-	no
Ex_8	a	b-	c-	d-	no

Let θ=0.2, then 5 patterns with Class- constraints are discovered as illustrated in Table 2. Length of each pattern is no less than 2. Class value is included in each pattern. Itemset Q_1=<a+, c+, yes> is a frequent pattern with Class-constraints. Q_1 appears in examples Ex_1, Ex_2, Ex_3, Ex_5, so $freq(Q_1) = 4$ and $support(Q_1) = 4/8 = 0.5 > 0.2$. There exists no parent itemest with support equal to Q_1. Itemset Q_2=<a+, yes> is not closed, for Q_1 is the parent with the same support.

Table 2. Closed frequent patterns with θ=0.2.

Pid	Pattern	Frequency
Q_1	< a+, c+, yes >	3
Q_2	< a+, c+, d+, yes >	2
Q_3	< a+, d-, no >	2
Q_4	< b-, c-, d-, no >	3
Q_5	< d-, no >	4

3. Algorithm PatHT

There are two main procedures to build classification algorithms based on patterns. First, discover closed frequent patterns with Class-constraint. Second, use patterns to train decision trees.

Bifet proposed three steps of data stream classification repeating cycle as shown in Figure 1[5]. In this paper, a novel cycle IPLmodel based on patterns is proposed. There are four steps as shown in Figure 2. Unlike in Figure 1, the training example is used to generate patterns at first, and then patterns are used to learn.

- *Step* 1. The algorithm processes a new training example Ex_{new} in a data stream.
- *Step* 2. The pattern mining algorithm processes Ex_{new} and gets its frequent pattern sets PS_{new}.
- *Step* 3. The classification algorithm processes PS_{new}, updating its data structures.
- *Step* 4. The algorithm is prepared to receive the new training example. Whenever necessary, it can predict or classify test examples.

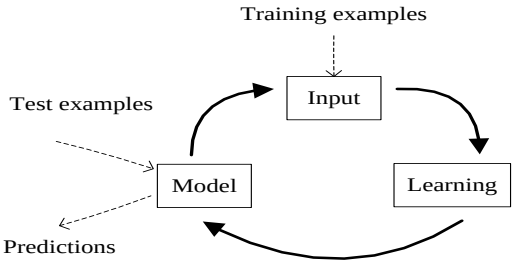


Figure 1. The data stream classification cycle [5].

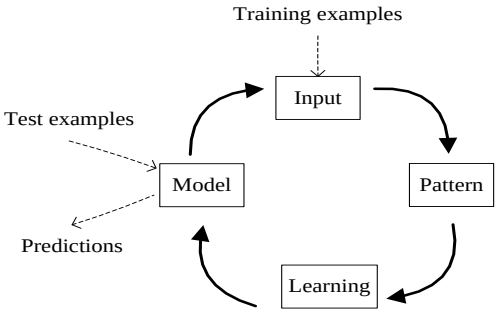


Figure 2. Novel data stream classification cycle IPLmodel.

Frequent pattern contains more information than a single attribute and may drop noise information. Because concept drift problem exists in some evolving data stream, frequent patterns are discovered in new examples. Therefore, pattern based methods can get more reasonable decision tree than common methods.

3.1. Closed Frequent Pattern Mining Algorithm

We propose algorithm Constrained and Closed Frequent Pattern Mining over data stream (CCFPM) to discover frequent patterns with Class- constraints. CCFPM includes two methods to deal with new examples and historical examples based on the SWM. Method CCFPMADD processes new example Ex_{new} and method CCFPMREMOVE processes old example Ex_{old}.

The data structures to deal with new example and old example are similar and the procedures are inversed. So, we mainly introduce the procedure CCFPMADD. There are three data structures *ClosedTable* [29], *CidList* [29] and *NewExample - Table/OldExampleTable* in CCFPM algorithm. Closed itemsets are maintained in *ClosedTable* which consists of three fields: *Pid*, *CP* and *SCP*. Each closed itemset *CP* is assigned a unique closed identifier *Pid*, and its frequency is denoted as *SCP*. When a new transaction Ex_{new} arrives, the frequencies of all itemsets associated with Ex_{new} in *Closed Table* are needed to be updated. *PidList* is used to maintain the information of each *item* in data stream and the related sets of *Pid*. *New Transaction Table*, which consists of two fields: *TempItem* and *Pid* is used to maintain the information of new example Ex_{new}. Variable *TempItem* is used to store the information of itemsets which satisfy condition $\{Ex_{new} \cap CP,\ CP \in ClosedTable\}$.

We store Ex_{new} to table *NewTransactionTable* at first after Ex_{new} arrived. Next, compare each *item* in Ex_{new} with *PidList* to update *New- TransactionTable*. Finally, add new patterns to *ClosedTable* or update existed patterns in *ClosedTable* referring to *NewTransactionTable*. Our algorithm processes each new example repeatedly to update data structures.

In this paper, we propose an incremental updated method to discover frequent patterns. Figure 3 illustrates the process to discover frequent patterns in the new sliding window. Each new example is used to update data structures about closed frequent patterns. And examples in the sliding window are denoted as B_i, and mined pattern sets are denoted as PS_i. That is, input data stream *DS* and output frequent patterns *PS*. Both *DS* and *PS* can be used as training examples. However, *PS* must be the latest patterns PS_{new} and historical patterns are dropped.

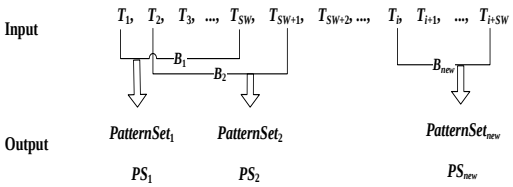


Figure 3. Process of mining frequent patterns.

Two kinds of training examples are used in the proposed algorithm, as shown in Figure 4. Use the pattern set *PS* as training examples only, or use *PS* and original data set *DS* together as training examples. Patterns used in the former are more than in the later. For the number of patterns is far less than a number of original training data, using *PS* only can significantly improve the efficiency of training. The aim is to get almost same accuracy as traditional classification. Using *PS* and *DS* together as training data will not significantly add time cost, and aim to get higher classification accuracy than traditional classification.

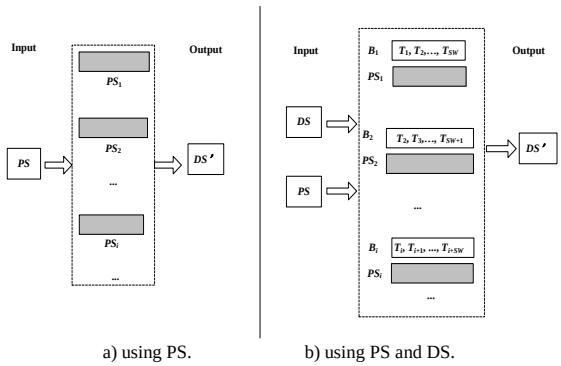


Figure 4. Two ways to use *PS* as training examples.

3.2. Classification Algorithm based on Patterns

When choosing the decision tree split tests, there are some popular and mature standards. Maybe the most ordinary is access to information gain. Therefore, we use information gain to build a pattern-based decision tree. The weight or frequency of pattern should be

considered. So, examples from the original data stream and discovered patterns should be modified. The process includes:

a. Adding weight '1' to each example, and the data set denotes as *DS*.
b. Adding missing values to each pattern and using frequency as weight, and the data set denotes as *PS*. In order to compare the process of selecting splitting nodes, some variables are used as shown in Table 3.

Table 3. Some variables.

Variable	Content
A_i	the *i*th attribute
A_{ij}	the *j*th value of A_i
C_k	the *k*th value of class label
WV_k	the total weights of C_k
WV_{ij}	the total weights of A_{ij}
WV_{ijk}	the total weights of A_{ij} under the condition of C_k
SumWV	the total weights of all examples in sliding window

- Example 2. Let's change from examples in Tables 1 and 2 to examples in Table 4 (DS1) and Table 5 (PS1). Weight of original data is 1 and weight of pattern is its frequency. And add missing values '?' to pattern. There are 8 examples in DS1 and 20 probability statistics of WV_{ijk} needed to be computed. While, there are 5 examples in PS1 and only 12 probability statistics of WV_{ijk} needed to be computed.

Table 4. Dataset *DS*1 with weights.

Example	A_1	A_2	A_3	A_4	Class	Weight
Ex_1	a+	b+	c+	d+	yes	1
Ex_2	a+	b+	c+	d-	yes	1
Ex_3	a+	b-	c+	d+	yes	1
Ex_4	a+	b-	c-	d-	no	1
Ex_5	a+	b	c+	d+	yes	1
Ex_6	a+	b	c+	d-	no	1
Ex_7	a-	b-	c-	d-	no	1
Ex_8	a	b-	c-	d-	no	1

Table 5. Pattern set PS1 with weights.

Example	A_1	A_2	A_3	A_4	Class	Weight
Ex_1	a+	?	c+	?	yes	3
Ex_2	a+	?	c+	d+	yes	2
Ex_3	a+	?	?	d-	no	2
Ex_4	?	b-	c-	d-	no	3
Ex_5	?	?	?	d-	no	4

From Tables 4 and 5, training on the pattern sets can reduce time consumption and generate more compact tree structure. These shortages of the two ways to build trees are followings, the former tree may be too big and may get overfitting problem; the later tree may have insufficient information and may get the lower classification accuracy. Therefore, when dealing with mass data, top-*k* frequent patterns and original data can be used together to learn a model. Advantage is that it can improve the efficiency of selecting splitting attribute and get high classification accuracy.

3.3. Novel Algorithm PatHT

A Novel way to build a decision tree based on frequent patterns is proposed in this paper. First, it incrementally updating generates closed frequent patterns with Class-constraints. The sliding window is used to mine patterns in new examples. Then, incrementally update decision tree using top-*k* closed frequent patterns and original data.

Five rules are used in novel algorithms:

- *Rule 1.* Only closed frequent pattern with class label are discovered. It means that at least one class value and one attribute value are included in a pattern.
- *Rule 2.* The top-k pattern is chosen if it contains most or all of distinguishable values of class.
- *Rule 3.* Missing value is engaged in the statistics of *SumWV*.
- *Rule 4.* If the difference in observed information gain is more than ε, then split on the best attribute. Else if *Gain*(best attribute) - *Gain*(second best attribute) < ε, then select the attribute with more total weights (missing values are not included) as the best attribute.
- *Rule 5.* If the total weights of the two best attributes are still same, then choose the first one as the split attribute.

The novel algorithm Pattern-based Hoeffding Tree (PatHT) includes three parts. Algorithm PatHT is the main function. Input parameters are S, SW, θ and δ. Output parameter is HT. Two kinds of training data PS only or PS and DS together are used as introduced before.

Algorithm PatHT()

Pattern-based Hoeffding Tree
Input: S data stream,
θ minimum support,
SW size of sliding window,
δ desired probability of choosing the correct attribute at any given node
Output: HT decision tree
1 For each transaction T_{new} in S Do
Get novel set of patterns PS_{new} = CCFPM(T_{new}, θ, SW);
Goto step 2 to 4
2 Let HT be a tree with a single leaf(root)
3 Initial counts WV_{ijk} at root
4 For each example (x, y, weight) in PS_{new} (and T_{new}) Do
HTreeGrow((x, y, weight), HT, δ)

Algorithm CCFPM is used to discover frequent patterns incremental updated. Function support() means support of itemsets. All patterns are satisfied class-constraints. That means that class and attribute must be contained in a pattern and length of a pattern is more than 2. The used three data structures are ClosedTable, CidList and NewTransactionTable.

Algorithm CCFPM()

Mining closed frequent patterns with constraints over data stream
Input: T_{new} new example,
θ minimum support,
SW size of sliding window,

Output: PS set of novel closed frequent patterns
1 Add T_{new} to NewTransactionTable
2 Let inters = $T_{new} \cap$ ClosedTable() according to PidList
3 Add inters to NewTransactionTable
4 For each TempItem in NewTransactionTable Do
If interS $\in$ ClosedTable
Then update support(interS) and add interS to PS_{new}.
Else if support(interS) $\geq \theta$
Then add <interS, support(interS)> to ClosedTable and PS_{new}.
5 If item $\in T_{new}$ And item is not in PidList
Then add item to PidList
6 Return PS_{new}.

Algorithm HTreeGrow trains decision tree incremental updated, and it is a Hoeffding tree with some improvements. It adds weights to training examples. Therefore:

1. Statistical information should consider weights of examples.
2. Choosing the best splitting node should consider the information of frequency. That is in Rule 4. Concept drift detector ADWIN[3] is used to find concept change. Function *G*() means information gain and function *MaxWeightAttr*() is used to find a attribute of higher total weight.

Algorithm HTreeGrow()

Growth of Hoeffding Tree
Input: example (x, y, weight)
HT decision tree
δ desired probability of choosing the correct attribute at any given node
Output: HT decision tree
1 Sort (x, y, weight) to leaf l using HT
2 Update counts WV_{ijk} with weight at leaf l
3 Compute information gain G for each attribute from counts WV_{ijk}
4 Split leaf
4.1 If G(Best Attr. BA_1)-G(2nd best Attr. BA_2) > ε
Then let BA_1 to be best attribute BA
4.2 Else let best attribute
BA = MaxWeightAttr(Best Attr., 2nd best Attr.)
4.3 Split leaf l on BA
5 For each branch Do
5.1 Start new leaf l and initialize estimators ADWIN
5.2 If ADWIN has detected change
Then create a subtree st
5.3 If no sunbtree
Then st as a new subtree
Else if st is more accurate
Then replace current node with st

4. Experimental Analysis

4.1. Data Streams

In order to compare the pros and cons of different methods, real and synthetic data streams are used in this paper. The real one is Poker-hand from UCI [13]. It consists of 1,000,000 instances and 11 attributes. Synthetic evolving data stream SEA is from MOA [19]. It consists of 50,000 instances and 4 attributes. In order to discover patterns, numeric attributes of SEA are discretized by method Discretize() in WEKA[28]. Synthetic data stream LED is generated by using generators in MOA [19]. About 1,000,000 examples of LED are generated. In order to analyze the ability of algorithm PatHT, two kinds of LEDs are generated in this paper: with and without concept drift. Method ConceptDriftStream [19] in MOA is used to set different widths of concept change W. And W is set less than or more than size of the sliding window.

In this paper, we compare the pros and cons of the algorithms NB, VFDT [10], HAT [21], HOT [2], AdoHOT [27], ASHT [27], OzaBoost [3], OzaBagAdwin [3] with PatHT. Method *EvaluatePrequential* [19] in MOA is used to test the performances of these algorithms by testing then training each example in sequence.

4.2. Performance

At first, analyze the discovered frequent patterns on data streams. Let the sliding window size SW = 1000 which is as same as size of window in method EvaluatePrequential [19] and the common size of the sliding window used in frequent pattern mining algorithms. In order to explain the changing trend of patterns, 5,000 examples are divided into 5 blocks {B_1, B_2, B_3, B_4, B_5}. Size of each block (denoted as BS) is equal to size of sliding window. Then, mining closed frequent patterns with class-constraints on each block.

Table 6 shows the frequent patterns on stable data stream Poker-hand, evolving data streams SEA and LED. At the same minimum support threshold, some conclusions can be got as following:

1. Discovered frequent patterns on Poker- hand are the most among three data streams. And the average length and maximal frequency of patterns are similar to each other of five blocks as shown in Table 6. The length of pattern in each block is from 2 to 4 and the average length is 2.83. Length of example in the data stream is 11, so the length of pattern to example ratio is 1 to 3.9 (2.83:11). The number of examples in each block is 1000 and the average number of patterns is 300, so the number of pattern to example ratio is 1 to 3.3 (300:1000).
2. Discovered patterns on SEA are less than on other two data streams, as shown in Table 6. The length of pattern in each block is from 2 to 3 and the average length is 2.3. The average number of patterns is about 75, so the number of pattern to example ratio is 1 to 17.5. Therefore, compared with the number of examples in each block, the number of patterns is small.
3. When the width of change W is set to 500, a number of patterns in five blocks on LED are different from each other as shown in Table 6. For example, the number of pattern in B_1 is 17 and the number in B_5 is 264 particularly. Besides number, length and frequency of patterns in LED change obviously

compared with other two data streams. There are two reasons for this. One is the feature of LED and another is that the width of change is less than size of the sliding window. So, the concept drift problem cannot be handled by the sliding window. The average length of pattern on LED is 7.46, so the length of pattern to example ratio is 1 to 3.5 (7.46:25). The average number of patterns is 131, so the number of pattern to example ratio is 1 to 7.6.

4. When $W < SW$, the distribution of pattern lengths on LED is shown in Figure 5. It can get that the distributions of patterns in B_1 is very different from these patterns in B_5. The numbers in B_5 and B_3 are more than numbers in other blocks. Numbers in B_4 and B_2 are in the middle of the five blocks, and the number in B_1 is the lowest. Overall, the distributions of patterns vary widely from different blocks.
5. Table 7 shows the ratios of pattern to example. Three data streams with different features have diverse ratios. Such as, patterns on Poker are far more than patterns on SEA. Compared with the length of original example, the length of pattern in LED is the shortest relatively and the length of pattern in SEA is the longest relatively.

Table 6. Pattern sets on data streams.

	Poker-hand			SEA			LED(*W*=500)		
	#P	Average length	Max weight	#P	Average length	Max weight	#P	Average length	Max weight
B_1	330	2.87	142	45	2.27	92	17	6.94	47
B_2	277	2.83	148	64	2.33	107	122	7.11	79
B_3	296	2.82	140	61	2.33	96	161	7.84	83
B_4	277	2.81	139	57	2.32	96	93	7.15	81
B_5	320	2.82	140	59	2.25	111	264	8.27	88

Table 7. The length ratio and number ratio of pattern to example.

	Length ratio	Number ratio
Pocker-hand	1 : 3.9	1 : 3.3
LED	1 : 3.5	1 : 7.6
SEA	1 : 1.7	1 : 17.5

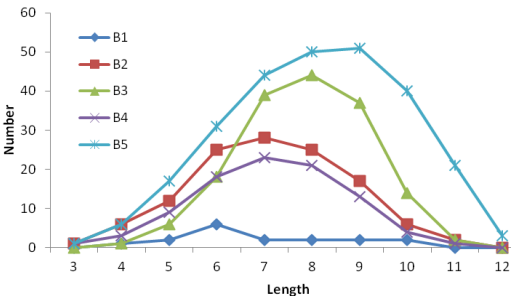
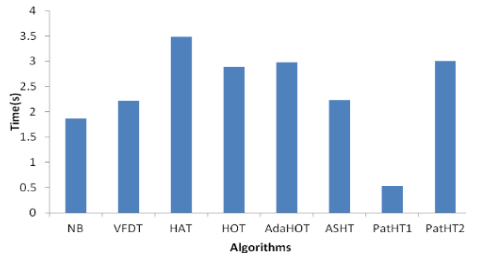


Figure 5. Distribution of patterns of LED (*W*=500).

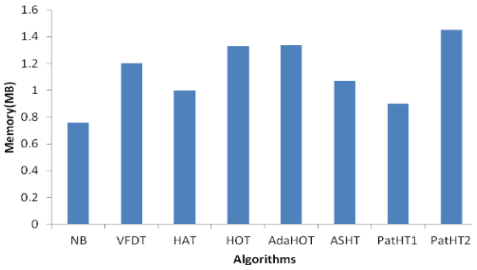
The second experiment is used to analyze the performances of 7 algorithms on real data stream Poker-hand. The comparisons of time cost, memory cost, and accuracy are shown in Figure 6. There are two kinds of training examples. One is only pattern sets used in PatHT1, and another is the combination of pattern sets and original examples used in PatHT2. The top-*k* patterns are used in algorithm PatHT, and the number of patterns in PatHT1 (about 30% of examples) is more than in PatHT2 (about 20% of examples).

The classification accuracy of algorithm PatHT1 is about 4% lower than the average accuracy of other six algorithms, as illustrated in Figure 6-c. But the training time cost of PatHT1 is much less than others, that is about 80% time is reduced as shown in Figure 6-a. This is because the number of training patterns is about 30% of original training examples from data streams. Top-*k* patterns and original examples are used as training data in PatHT2. From Figure 6, it can get that accuracy of PatHT2 is about 20% more than the average accuracy of other algorithms. For the number of patterns is very small, the time cost and memory cost are little more than other algorithms. But additional time and memory costs are used to discover patterns.

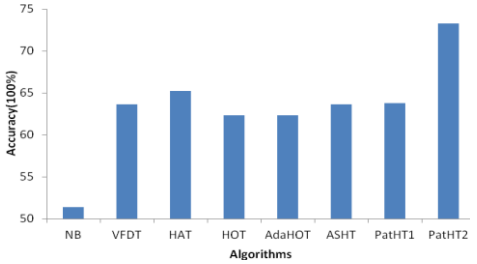
Last experiment is used to compare the performances of different algorithms on data stream SEA and LED. The width of change in SEA is uncertain. There are three kinds of LEDs with no drift, width of change W=500 (less than SW) and W=2000 (more than SW). Table 8 shows the accuracies of nine algorithms on data stream SEA and LED.



a) time.



b) memory.



c) accuracy.

Figure 6. Performances of algorithms on Poker-hand.

Table 8. Comparisons of accuracies on evolving data streams SEA and LED.

	SEA			**LED**			**LED**			**LED**		
	Drift			**No Drift**			***W*=500**			***W*=2,000**		
	Time	**Acc.**	**Mem.**	**Time**	**Acc.**	**Mem.**	**Time**	**Acc.**	**Mem.**	**Time**	**Acc.**	Mem.
NB	1.01	82.8	0.76	3.88	72.1	1.28	3.31	73.2	1.28	3.57	73.5	0.80
VFDT	1.47	86.3	0.99	6.18	72.4	0.98	5.43	73.1	1.06	5.63	74.7	1.42
HAT	2.62	90.4	0.99	9.30	71.3	0.75	8.64	72.1	1.06	9.20	73.9	1.12
HOT50	2.39	86	2.1	6.91	72.7	0.92	6.63	73.5	1.47	8.13	74.4	1.10
AdoHOT5	2.34	86	2.1	6.93	72.7	0.68	6.61	73.5	1.49	8.52	74.3	1.13
ASHT	1.47	86.3	1.01	5.69	72.4	1.07	5.48	73.1	1.34	6.16	74.7	1.31
OzaBagAdwin	7.78	90	2.41	53.98	72.6	1.91	54.65	**73.7**	2.10	54.58	74.5	2.22
OzaBoost	5.44	88.4	2.79	40.65	73.2	1.81	41.32	73.2	1.82	41.14	74.8	2.26
PatHT	2.3	**91.7**	2.6	7.80	**73.4**	1.31	6.81	71.9	0.75	7.23	**75.8**	1.51

From all the experiments above mentioned, the conclusions can be drawn as follows.

1. When dealing with the dense and stable data streams, such as real data stream Poker-hand, the accuracy of PatHT is same as other algorithms. But only frequent patterns are used as training data which are much less than original training examples. Therefore, compared with known algorithms, a lot of training time cost can be reduced.
2. When processing stable data streams, using combination training data of top-*k* patterns and original examples, PatHT algorithm can get better performance than other algorithms. The additional cost is using more time to train. The number of pattern is small, so the additional time cost is little. Because only high frequency patterns are used in training, the classification model is more compact. Therefore, the memory cost is lower than some known algorithms.
3. When processing evolving data streams, PatHT can get high accuracy over data streams with the big width of change (more than size of the sliding window). But the accuracy may be low when handling data streams with small width of change (less than size of sliding window).
4. Addition time and memory may be used in PatHT for mining frequent patterns.

5. Conclusions

A data stream classification repeating cycle common has three steps of input-training-model. There is some useless information in training data, such as noises. In order to drop some useless information, a novel classification repeating cycle IPLmodel is proposed in this paper: input - pattern - training - model. That is pattern mining on examples before training. Incremental updated pattern mining algorithm is used to discover closed frequent patterns with class-constraints. And these patterns are used as training data independently or as a part of combination. An evaluation study on real and synthetic data streams figures that compared with some famous methods, the new algorithm PatHT has better performance. The classification method based on patterns can get high classification accuracy or can reduce the training time significantly. The additional costs are the time and memory consumption of discovering frequent patterns. And in order to discover patterns, numeric attributes are required to be discretized.

References

[1] Assunção M., Veith A., and Buyya R., "Distributed Data Stream Processing and Edge Computing: A Survey on Resource Elasticity and Future Directions," *Journal of Network and Computer Applications*, vol. 103, pp. 1-17, 2018.

[2] Bifet A. and Gavaldá R., "Adaptive Learning from Evolving Data Streams," *in Proceedings of the 8th International Symposium on Intelligent Data Analysis*, Lyon, pp. 249-260, 2009.

[3] Bifet A. and Gavaldá R., "Learning from Time-Changing Data with Adaptive Windowing," *in Proceedings of the 7th SIAM International Conference on Data Mining*, Minnesota, pp. 443-448, 2007.

[4] Bifet A., Gavaldà R., Holmes G., and Pfahringer B., *Machine Learning for Data Streams with Practical Examples in MOA*, MIT Press, 2018.

[5] Bifet A., Holmes G., and Pfahringer B., "New Ensemble Methods for Evolving Data Streams," *in Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Paris, pp. 139-148, 2009.

[6] Bifet A., Zhang J., and Fan W., He C., Zhang J., Qian J., Holmes G., and Pfahringer B., "Extremely Fast Decision Tree Mining for Evolving Data Streams," *in Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Halifax, pp. 1733-1742, 2017.

[7] Bustio-Martínez L., Cumplido R., Hernández-León R., Bande-Serrano J., and Feregrino-Uribe C., "On the Design of Hardware Software Architectures for Frequent Itemsets Mining on Data Streams," *Journal of Intelligent Information Systems*, vol. 50, no. 3, pp. 415-440, 2018.

[8] Chen H., Shu L., Xia J., and Deng Q., "Mining Frequent Patterns in A Varying-Size Sliding Window of Online Transactional Data Streams,"

Information Sciences, vol. 215, no. 12, pp. 15-36, 2012.
[9] Chi Y., Wang H., Yu P., and Muntz R., "Catch the Moment: Maintaining Closed Frequent Itemsets over A Data Stream Sliding Window," *Knowledge and Information Systems*, vol. 10, no. 3, pp. 265-294, 2006.
[10] Domingos P. and Hulten G., "Mining High-Speed Data Streams," *in Proceeding of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Boston, pp. 71-80, 2000.
[11] Farhat A., Gouider M., and Said L., "New Algorithm for Frequent Itemsets Mining From Evidential Data Streams," *Procedia Computer Science*, vol. 96, pp. 645-653, 2016.
[12] Farid D., Zhang L., Hossain A., Rahman C., Strachan R., Sexton G., and Dahal K., "An Adaptive Ensemble Classifier for Mining Concept Drifting Data Streams," *Expert Systems with Applications*, vol. 40, no. 15, pp. 5895-5906, 2013.
[13] Frank A. and Asuncion A., UCI Machine Learning Repository [EB/OL]. Irvine, CA: University of California, School of Information and Computer Science, http:// archive.ics.uci.edu/ml, Last Visited, 2010.
[14] Gama J., Fernandes R., and Rocha R., "Decision Trees for Mining Data Streams," *Intelligent Data Analysis*, vol. 10, no. 1, pp. 23-45, 2006.
[15] Gama J. and Medas P., "Learning Decision Trees from Dynamic Data Streams," *Acm Symposium on Applied Computing*, vol. 11, no. 8, pp. 1353-1366, 2005.
[16] Gomes H., Barddal M., Enembreck F., and Bifet A., "A Survey on Ensemble Learning for Data Stream Classification," *ACM Computing Surveys*, vol. 50, no. 2, pp. 1-23, 2017.
[17] Hammer H., Yazidi A., and Oommen B., "On the Classification of Dynamical Data Streams Using Novel "Anti-Bayesian" Technique," *Pattern Recognition*, vol. 76, pp. 108-124, 2018.
[18] Han M., Ding J., and Li J., "TDMCS: An Efficient Method for Mining Closed Frequent Patterns Over Data Streams Based on Time Decay Model," *The International Arab Journal of Information Technology*, vol. 14, no. 6, pp. 851-860, 2017.
[19] Holmes G., Kirkby R., and Pfahringer B., MOA: Massive Online Analysis. http://sourceforge.net/ projects/moa-datastream. Last Visited, 2014.
[20] Hulten G, Spencer L., and Domingos P., "Mining Time-Changing Data Streams," *in Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, pp. 97-106, 2001.
[21] Kourtellis M., Morales G., Bifet A., and Murdopo A., "VHT: Vertical Hoeffding Tree," *in Proceedings of the IEEE International Conference on Big Data*, Washington, pp. 915-922, 2016.
[22] Kusumakumari V., Sherigar D., Chandran R., and Patil N., "Frequent Pattern Mining on Stream Data Using Hadoop Cantreegtree," *Procedia Computer Science*, vol. 115, pp. 266-273, 2017.
[23] Lim Y. and Kang U., "Time-Weighted Counting for Recently Frequent Pattern Mining in Data Streams," *Knowledge and Information Systems*, vol. 53, no. 2, pp. 391-422, 2017.
[24] Mello R., Vaz Y., Grossi C., and Bifet A., "On Learning Guarantees to Unsupervised Concept Drift Detection on Data Streams," *Expert Systems with Applications*, vol. 117, pp. 90-102, 2019.
[25] Oza N. and Russell S., *Online Bagging and Boosting*. Morgan Kaufmann, 2001.
[26] Reddy V., Rao T., and Govardhan A., "Fast and Lossless Mining of Closed Frequent Itemsets Over Data Streams," *Journal of Advanced Research in Dynamical and Control Systrems*, vol. 1, pp. 256-263, 2018.
[27] Wang P., Wu X., Wang C., Wang W., and Shi B., "CAPE-A Classification Algorithm Using Frequent Patterns Over Data Streams," *Journal of Computer Research and Development*, vol. 41, no. 10, pp. 1677-1683, 2004.
[28] Witten I. and Frank E., *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, 2005.
[29] Yen S., Lee Y., Wu C., and Lin C., "An Efficient Algorithm for Maintaining Frequent Closed Itemsets Over Data Stream," *Next-Generation Applied Intelligence*, vol. 5579, no. 1, pp. 767-776, 2009.
[30] Yen S., Wu C., Lee Y., Tseng V., and Hsieh C., "A Fast Algorithm for Mining Frequent Closed Itemsets Over Stream Sliding Window," *in Proceedings of IEEE International Conference on Fuzzy Systems*, Taipei, pp. 996-1002, 2011.

Han Meng born in 1982, Ph.D., associate professor. Her research interests include data mining and machine learning.

Jian Ding born in 1977, M.S., associate professor. His research interests include machine learning and data mining.