A novel data structure for efficient representation of large data sets in data mining

#Radhika M. Pai , Ananthanarayana V.S.
National Institute of Technology Karnataka, Srinivasanagar P.O, Karnataka-575025, India
e-mail:radhikampai@rediffmail.com, anvs@nitk.ac.in

***Abstract*--An important goal in data mining is to generate an abstraction of the data. Such an abstraction helps in reducing the time and space requirements of the overall decision making process. It is also important that the abstraction be generated from the data in small number of scans. In this paper, we propose a novel data structure called Prefix-Postfix structure(PP-structure), which is an abstraction of the data that can be built by scanning the database only once. We prove that this structure is compact, complete and incremental and therefore is suitable to represent dynamic databases. Further, we propose a clustering algorithm using this structure. The proposed algorithm is tested on different real world datasets and is shown that the algorithm is both space efficient and time efficient for large datasets without sacrificing for the accuracy. We compare our algorithm with other algorithms and show the effectiveness of our algorithm.**

***Index terms*-- Prefix-Postfix structure, PC-tree, PPC-tree, Clustering, Data Mining, Data structure, incremental algorithm, dynamic databases**

I. INTRODUCTION

Clustering is an exploratory data analysis task and has been widely applied in many areas such as pattern recognition and image processing, information processing, medicine, geographical data processing, and so on. Most of these domains deal with massive collections of data. In data mining applications, both the number of patterns and features are typically large and cannot be stored in main memory and hence needs to be transferred from secondary storage as and when required. This takes a lot of time. In order to reduce the time, it is necessary to devise efficient algorithms to minimize the disk I/O operations. Hence, the methods to handle them must be efficient in terms of data set scans and memory usage. Several algorithms have been proposed in the literature for clustering large data sets[2,3,4,11].

Most of these algorithms need more than one scan of the database. To reduce the number of scans and hence the time, the data from the secondary storage are stored in main memory using abstractions and the algorithms access these data abstractions and hence reduce the disk scans. Some abstractions to mention are the CF-tree[4], FP-tree[4], PC-tree[8], PPC-tree[6], kd-trees[5], AD-trees[1]. The CF–tree[4] is the cluster feature vector tree which stores information about cluster descriptions at each node. This tree is used for clustering. The construction of this tree requires a single scan provided the two factors B and T are chosen properly. The FP-tree[4] is used for association rule mining and stores crucial and quantitative information about frequent sets. The construction of this tree requires two database scans and the tree is dependent on the order of the transactions. The kd-tree[5] and the AD-trees[1] reduce the storage space of the transactions by storing only the prototypes in the main memory. These structures are well suited for the applications for which they have been developed, but the use of these structures is limited as it is not possible to get back the original transactions. i.e. the structure is not a complete representation. PC-tree[8] is one such structure which is order independent , complete and compact. By using this structure, it is possible to retrieve back the transactions. The PC-tree[8] is a compact structure and the compactness is achieved by sharing the branches of the tree for the transactions having the same prefix. The tree generates new branches if the prefixes are different. One more abstraction called PPC-tree[6] is similar to PC-tree but it partitions the database vertically and constructs the PC-tree for each partition separately. The drawback of this structure is that, it is not possible to retrieve back the original transactions and the use of this structure in clustering is very much dependent on the partitioning criteria. The advantage of this structure is that it generates some synthetic patterns useful for supervised clustering. Both these abstractions need only a single database scan.

The problem with the PPC-tree is that, by looking at the data set, it is difficult to predict the number of partitions in advance. To overcome this, we came up with a new structure called an Offset-tree which stores the first transaction as a linked list with all the features present, (the node structure being same as the PC-tree) and thereafter for subsequent transactions, the count field is incremented if the features are

1-4244-0716-8/06/$20.00 ©2006 IEEE.

same or else only the offset is stored as a sibling to the node. This tree is very much compact than the PC-tree or the PPC-tree. But, the problem with this structure is that, when used in clustering, the accuracy drops down due to overtraining. The failure of this structure inspired us to think of a structure which lies in between the Offset-tree and the PC-tree, and the result is this new structure which we call the Prefix-postfix-structure. This name is given because the transactions which have the same prefix or the same postfix have their nodes being shared by both. This results in quite compact representation and this doesn't have the problem of fixing the number of partitions as in the case of the PPC-Tree.

In this paper, we propose a novel data structure which has all the properties of the PC-tree and is still compact than the PC-tree. The compactness is achieved by sharing the branches of the tree for the transactions having the same prefix as well as the same suffix. This type of transactions in which the prefix or the suffix being same with other transactions occur naturally in the handwritten digit dataset.

For e.g. Consider the patterns which represent the digit 2 as given in Figure 1. For simplicity , we just show a 4x4 matrix for a pattern. But the real world datasets have orders which are very larger than this for each pattern.

In the patterns given in Figure 1, the empty cells are considered as zeroes. By taking the positional values of the cells having ones, each pattern can be represented as a feature list as follows.

Pattern #1: 1,2,3,7,10,13,14,15
Pattern #2: 0,1,2,3,7,9,10,12,13,14,15
Pattern #3: 1,2,3,5,7,10,13,14,15
Pattern #4 : 1,2,3,4,7,10,12,13,14,15

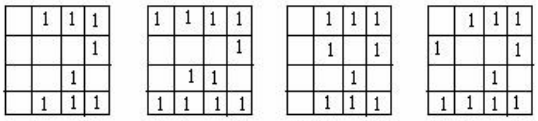

Figure 1. Patterns for digit 2

In the PC-tree, if two transactions have different prefixes as for patterns numbered 1 & 2, the algorithm results in 2 separate branches, even though the two transactions have the same suffix. In our structure, we propose to reduce the space further, by sharing the branches of the transactions having the same suffix also.

Thus for patterns 1 and 2, the initial branch will be different but later will have their branches merged. Further advantage of this structure is that it generates some synthetic patterns which help in increasing the accuracy of clustering algorithm. The advantage of this structure is that using this structure one can get back the original transactions as required by some other applications as well as generate some synthetic patterns not present in the original transactions which can aid in clustering.

The paper is organized as follows. The novel structure, the PP-structure, is described in Section 2, the experimental results are discussed in section 3 and section 4 gives the conclusion.

II. PREFIX–POSTFIX STRUCTURE(PP-STRUCTURE)

The PP-structure is an abstract and compact representation of the transaction database. It is also a complete representation, order independent and incremental.

Let I={i1,i2,.....in} be a set of items. A transaction is a non-empty ordered subset of I.

A pattern is a nonempty ordered subset of a transaction.

We consider both transactions and patterns as ordered subsets where item numbers appear in an increasing order.

A prefix-postfix structure stores all transactions of the database in a compact way.

Each node of the PP-structure consists of four fields.

They are

'Feature' specifies the feature value of a pattern.

'Count' is a signed field. The absolute value specifies the number of patterns represented by a portion of the path reaching this node. The sign bit being 1 indicates that the current branch is attached to an existing branch because of the same suffix. This portion differs from that of the PC-tree.

'Child_pointer' represents the pointer to the following path.

'Sibling_pointer' points to the node which indicates the subsequent other paths from the node under consideration.

Figure 2 shows the node format of a PP- structure.

Feature	Count	Child_pointer	Sibling_pointer

Figure 2. Node structure of the PP-structure

A. Construction of the PP-structure

The algorithm for the construction of the PP-structure is as follows.

Let T_r be the transaction database

Let root of the PP-structure be T

For each pattern, $t_i \in T_r$

Let m_i be the set of feature values in t_i

If no sub-pattern starting from T (prefix pattern) exists corresponding to m_i then

create a new branch, with a node having 'feature' fields as first feature value of m_i and count field with value set to 1.

else

put the values of m_i in an existing branch e_b by incrementing corresponding absolute count field values of the nodes in e_b . When there is no match, create a new branch with node having feature value as the first mismatched feature value of m_i and count value as 1.

Search the PP-structure already constructed and find a maximal branch x_b of the PP-structure whose nodes match with the maximum length suffix of the values in m_i . let m_{ik} be the feature value in m_i from where the suffix matches with the feature values of the in x_b . For

the feature values from current position to m_{ik} , create new nodes with feature values equal to values in the set m_i and count field equal to1. Attach it to x_b and increment the count field of x_b by 1. Set the count value of the node which is attached to e_b as –count.

If no branch x_b is found, create new nodes with feature values equal to the feature values in m_i and count field set to 1 till the end of set m_i.

Retrieval.

For each branch of the PP-structure,

Traverse the branch and print the feature value of each node and decrement the count value.

Repeat the above step for all branches.

B. Comparison of PP-structure and PC-tree

The PC-tree[8] compacts the database by merging the nodes of the patterns having the same prefix. But in the PP-structure, still compaction is achieved by merging the branches having the same suffix also and so the number of nodes is reduced thus saving considerable space.

An example: Consider the following set of transactions as shown in Figure 3(a). The first column gives the transaction number, the second column gives the set of features and the last column gives the label. The PP-structure and the PC-tree for the set of transactions of Figure 3(a) is given in Figure 3(b) and Figure 3(c) respectively. In the figures, the nodes are indicated by circles, the right arrow is the child pointer, the downward pointer is the sibling pointer. The first number inside the circle is the feature value and the number after the colon is the count. The last node in all branches is the label node.

.

Trans.#	Features	Label
1	1,2, 3, 4, 5, 8, 9,10,11,12,14,15	0
2	1,2,3,4,7,10,11,12,14,15	0
3	2,3 ,4,5, 6,12, 14,15	0
4	2, 4,5,7,9,12,13,14	3
5	2,4,5,6,8,12,13,14	3

Figure 3(a). Sample set of transactions

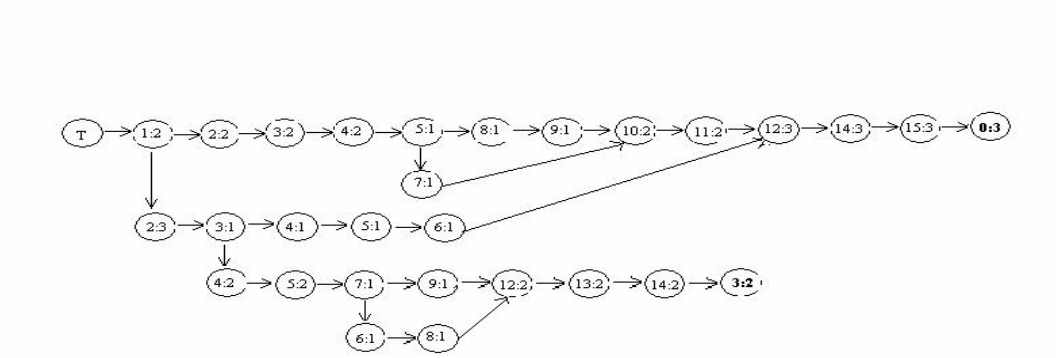

Figure. 3(b) PP-structure constructed for the set of transactions in Figure 3(a)

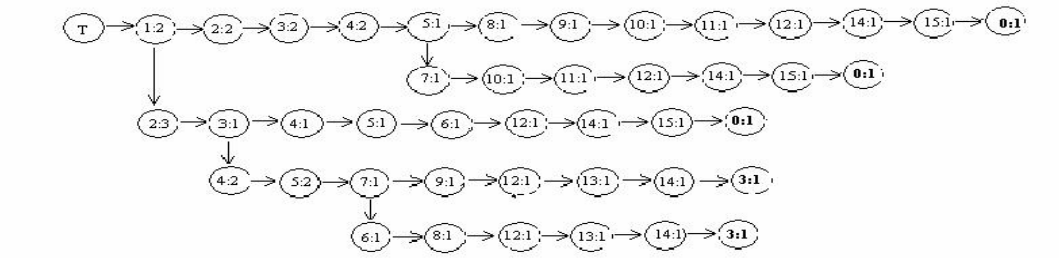

Figure. 3(c) PC-tree for the set of transactions in Figure 3(a)

From the figures, we see that the PP-tree requires 30 nodes whereas the PC-tree requires 44 nodes, the node structure being same in both the cases.

C. PP-structure based clustering algorithm

Let c be the number of classes.

For each branch bj in the PP-structure T

Find the matches between the test pattern and the branch bj. let it be C^j

Find k largest counts in decreasing order. Let them be C^1, C^2, … , C^k and let the corresponding labels be o_1, o_2, … o_k.

For p=1 to k

Compute the weight , $w_p = 1 - (C^p - C^l) / (C^k - C^p)$

For n = 1 to c

$Sum_n = \sum^k_{m=1}(w_m)$ where $(o_m == n)$

Output (label = o_x) for which Sum_x is maximum for x $\in$ {1,2,…c}

III. EXPERIMENTS AND RESULTS

To evaluate the performance of our algorithm, the following three real world datasets are considered.

A. Dataset 1: OCR data

This is a handwritten digit dataset. There are 6670 patterns in the training set, 3333 patterns in the test set and 10 classes.[6,7,8,12]. Each class has approximately 670 training patterns and 333 test patterns. Each pattern represents a digit which is in the binary matrix form of order 16 x 12. The binary matrix form of the digit is treated as transactions by considering the positional value of the features having value as 1. Thus each handwritten digit pattern has a maximum of 192 features.

We conducted experiments separately with 2000(200 patterns from each class), 4000(400 patterns from each class) and 6670 training patterns(667 patterns from each class) and compared our results with PC-tree based clustering algorithm, the PPC-tree based algorithm and the k-NNC algorithm. In PPC-tree based algorithm we chose the number of partitions as 4. In all the cases the k-value was chosen as 15. The results are summarized in Table-1.

Table–1 shows the comparison of PP-structure based clustering algorithm, the PPC-tree based algorithm and the PC-tree based clustering algorithm. For completeness, we have also compared our algorithm with the k-NNC algorithm.

Table-2 shows the results of various algorithms on this data set.

B. Dataset 2 : USPS data

This is a handwritten digit dataset which is a collection of handwritten digits scanned from the U.S. postal services[9]. There are 7291 patterns in the training set and 2007 patterns in the testing set and 10 classes. Each pattern represents a digit and has 256 features which is in the form of a 16x16 matrix, each having a feature value as a normalized real number in the range [-1..+1]. This pattern is converted into binary matrix by taking the feature with a value –1 as 0 and the remaining as 1. Again this matrix of features is converted to a transaction by taking the positional value of the features which have a value 1. We have compared our algorithm with the PC-tree based algorithm, the PPC-tree based algorithm with four partitions and the k-NNC algorithm. In all the algorithms , we have chosen the value of k as 5.

C. Data set 3: MNIST data

This is a data which is a mixture of the NIST(National Institute of standards and technology) special database 3 and 1[10]. This is collection of handwritten digits written by census bureau employees and high school students. This is a large data set having 60000 patterns in the training set and 10000 patterns in the test set and 10 classes. Each pattern is in the form of a matrix of order 28x28. Each feature has a value in the range 0 to 256. The value of 0 corresponds to the background gray value and the other corresponds to the foreground value. This matrix is converted to a binary matrix by taking the background pixel value as 0 and foreground pixel value as 1. This binary matrix is converted into a transaction by taking the positional value of the feature having a value 1. We have compared our algorithm with the PC-tree based algorithm and the PPC-tree based algorithm with 4 partitions.. We have not run the k-NNC algorithm as this is very expensive because of the large size of the data set. The data set can't be stored in the main memory and requires more than one data set scan.

All the experiments are executed on Xeon processor based Dell workstation precision 670 series having a clock frequency of 3.2 GHZ and 1 GB RAM. In each case, the classification accuracy , the memory space used, and the time (which includes both the design time and the testing time) is recorded. The results are tabulated in Table-1.

From the results we see that for OCR data, PP-structure based algorithm gives best results in terms of accuracy and space for 6670 transactions. The number of partitions fixed for the PPC-tree based algorithm for this dataset is 4.

For USPS data set, we observe that the PP-structure based algorithm gives good results in terms of accuracy. Though the k-NNC algorithm consumes less space than the PP-structure for this data set, the k-NNC algorithm fails for large data sets as the data set cannot be fitted in the main memory and the algorithm requires more than one data set scan. For this data set, we have fixed the number of partitions in the PPC-tree based algorithm as 4.

For the MNIST data set, we observe that the PP-structure outperforms the other algorithms with respect to time, space and accuracy. Here, we have fixed the number of partitions in PPC-tree based algorithm as 4.

From these experiments, it is evident that the PP-structure is an efficient abstraction for large datasets. The clustering algorithm based on this structure is efficient in terms of time, and space without sacrificing for the accuracy.

IV. CONCLUSION

In this paper, a novel data structure called Prefix-Postfix structure is proposed which stores the transactions of a transaction database in a compact way. This structure is

complete, order independent and incremental and is suitable to represent dynamic databases. The use of this structure in clustering is also proposed and the effectiveness of the algorithm for large data sets is established by comparing our algorithm with other algorithms such as the PC-tree based algorithm, the PPC-tree based algorithm and the k-NNC algorithm. The PP-structure based algorithm is found to outperform other algorithms for large data sets in terms of time, space and accuracy.

The performance of the algorithm is evaluated by testing the algorithm with 3 different datasets of handwritten digits and the effectiveness of our algorithm is thus established.

REFERENCES

[1] Andrew Moore, Mary Soon Lee, (1998), "Cached sufficient statistics for efficient machine learning with large datasets", *Journal of Artificial Intelligence Research* 8 (1998), pages 67-91.

[2] Anil K.Jain, Richard C.Dubes (1988), "Algorithms for Clustering Data", Prentice Hall Advanced Reference Series.

[3] A.K.Jain, M.N.Murty, P.J.Flynn(1999), "Data Clustering: A Review", *ACM Computing Surveys*, vol. 31, No.3, September 1999, pages 264-323.

[4]Arun K, Pujari(2001), "Data Mining techniques", University Press 2001.

[5] Friedman J.H., Bentley J.L., Finkel R.A.(1997), "An algorithm for finding best matches in logarithmic expected time",*ACM trans. Math software* 3 (3), pages 209-226

[6]P.Viswanath, M.N.Murthy(2002), "An incremental mining algorithm for compact realization of prototypes", *Technical Report, IISC, Bangalore*.

[7]M.Prakash, M.Narasimha Murthy(1997), "Growing subspace pattern recognition methods and their neural network models, IEEE trans. Neural Networks 8(1) 161-168.

[8]V.S.Ananthanarayana, M.Narasimha Murty, D.K.Subramanian(2003), "Tree structure for efficient data mining using rough sets", *Pattern Recognition Letters,* vol.24(2003),pages851-86.

[9] http://www.cs.cmu.edu/~15781/web/digits.html

[10]http://wwwi6.informatik.rwthaachen.de/~keysers/usps.html

[11] R.O.Duda , P.E.Hart (1973), "Pattern Classification and Scene Analysis", Wiley, New York.

[12]T.Ravindra Babu, M.Narasimha Murthy(2001), "Comparison of Genetic Algorithms based prototype selection scheme", *Pattern Recognition* 34 (2001), pages523-525.

Table 1. Comparison of PP-tree, PC-tree, PPC-tree and k-NNC based Algorithms

Expt. No.	Dataset	Algorithm	Storage space (in bytes) to store training patterns	Time in secs. (Training time + testing time)	Accuracy
1	OCR data (2000 patterns)	**PP-structure based algorithm**	**1123696**	**34**	**89.56**
		PC-tree based algorithm	1406528	153	89.56
		PPC-tree based algorithm(with 4 partitions)*	1119280	131	41.58
		k- NNC	1544000	18	89.44
2	OCR data (4000 patterns)	**PP-structure based algorithm**	**2070112**	**110**	**92.04**
		PC-tree based algorithm	2717792	245	91.96
		PPC-tree based algorithm(with 4 partitions)*	1970304	198	41.01
		k-NNC	3088000	31	91.90
3	OCR data (6670 patterns)	**PP-structure based algorithm**	**3216400**	**863**	**93. 76**
		PC-tree based algorithm	4408256	386	93.61
		PPC-tree based algorithm(with 4 partitions)*	3812320	314	64.3
		k-NNC	5149240	47	93.55
4	USPS data	**PP-structure**	**7991504**	**554**	**93.27**

		based algorithm			
		PC-tree based algorithm	10030656	537	92.68
		PPC-tree based algorithm(with 4 partitions)*	6490336	337	92.23
		k-NNC	7495148	39	93.27
5	MNIST data	**PP-structure based algorithm**	**108528480**	**5996**	**96.5**
		PC-tree based algorithm	126535296	28451	96.5
		PPC-tree based algorithm(with 4 partitions)*	77355312	17182	68.3
		k-NNC	188400000	cannot be stored in main memory	Not Implemented as it is expensive

* For the PPC-tree based algorithms, we have chosen the number of partitions as 4 in all the experiments. The accuracy, time and storage space varies depending on the number of partitions.

Table 2. Results of various classifiers on the OCR data set

Algorithm	**Accuracy(%)**
Leaders[12]	92.9
Medoids[12]	91.18
k-NNC[11]	93.55
GSM[7]	90.8
EALSM[7]	92.0
ALSM[7]	91.1
GLSM[7]	91.0
MLP[7]	90.4
PC-tree based[8]	93.61
PPC-tree based algorithm with 4 partitions[6]	64.3
PP-structure based	**93.76**