Prefix-Suffix Trees: A Novel Scheme for Compact Representation of Large Datasets

Radhika M. Pai[1] and V.S Ananthanarayana[2]

[1] Manipal Institute of Technology, Manipal
[2] National Institute of Technology Karnataka, Surathkal

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 scheme called Prefix-Suffix trees for compact storage of patterns in data mining, which forms an abstraction of the patterns, and which is generated from the data in a single scan. This abstraction takes less amount of space and hence forms a compact storage of patterns. Further, we propose a clustering algorithm based on this storage and prove experimentally that this type of storage reduces the space and time. This has been established by considering large data sets of handwritten numerals namely the OCR data, the MNIST data and the USPS data. The proposed algorithm is compared with other similar algorithms and the efficacy of our scheme is thus established.

Keywords: Data mining, Incremental mining, Clustering, Pattern-Count(PC) tree, Abstraction, Prefix-Suffix Trees.

1 Introduction

In today's technologically developing world, there is an increase in both collection and storage of data. With the increase in the amounts of data, the methods to handle them should be efficient in terms of computational resources like memory and processing time. In addition to this the database is dynamic. The state of database changes due to either addition/deletion of tuples to /from the database. So, there is a need for handling this situation incrementally, without accessing original data more than once for mining applications. Clustering is one such data mining activity which deals with large amounts of data. Clustering 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. They 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. Several algorithms have been proposed in the literature for

A. Ghosh, R.K. De, and S.K. Pal (Eds.): PReMI 2007, LNCS 4815, pp. 316–323, 2007.
© Springer-Verlag Berlin Heidelberg 2007

clustering large data sets[2,3,4]. 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, FP-tree[4], PC-tree[8], PPC-tree[6], kd-trees[5], AD-trees[1], the PP-structure[13].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 and for each class 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. The PP-structure[13] is similar to the PC-tree in the sense that it shares the branches for the transactions having the same prefix. But it differs from the PC-tree, in that it also shares the branches for the transactions having the same suffix. This structure also is a compact representation, but the construction of the structure is very complex as it searches the already constructed structure to check whether the current pattern's postfix is already present. In this paper, we propose a scheme called Prefix-Suffix Trees which is a variant of the PC-tree and which in principle is similar to the Prefix-Postfix structure in that the transactions having the same prefix or the same postfix have their branches being shared. The construction is simple compared to the Prefix-Postfix structure.The advantage of the Prefix-Suffix tree is that this also generates some synthetic patterns not present in the database which aids in clustering. The advantage of this scheme over the PPC-tree is that it is possible to get back the original transactions by storing a little extra information. In case of PPC-tree, if the number of parts is not selected properly, it results in overtraining and hence decrease in accuracy.

2 The Prefix-Suffix Trees Based Scheme

The Prefix-Suffix Trees based scheme which we propose is an abstract and compact representation of the transaction database. It is also a complete representation, order independent and incremental. The Prefix-Suffix Trees stores all transactions of the database in a compact way.

The Prefix-Suffix Trees are made up of nodes forming trees. Each node consists of four fields.

They are 'Feature' specifies the feature value of a pattern.The feature field of the last node indicates the transaction-id of the transaction which helps in retrieving the original transactions. 'Count' . The count value specifies the number of patterns represented by a portion of the path reaching this node. '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.

Fig.1 shows the node structure of Prefix-Suffix trees.

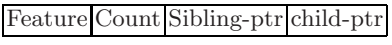

Fig. 1. Node Structure of the Prefix-Suffix Trees

2.1 Construction of the Prefix-Suffix Trees

The algorithm for the construction of the Prefix-Suffix Trees is as follows. Let T_r be the transaction database.

Partition the transaction database T_r into 2 equal parts. Let the 2 parts be T_{r1} and T_{r2} respectively.

For T_{r1} construct the tree as follows which is called Prefix Tree.

Let the root of the Prefix-tree be TR1.

For each pattern, $t_i \in T_{r1}$ Let m_i be the set of positions of non-zero values in t_i.

If no sub-pattern starting from TR1 exists corresponding to m_i,

THEN

Create a new branch with nodes having 'Feature' fields as values of mi and 'Count' fields with values set to 1.

ELSE

Put values of m_i in an existing branch e_b by incrementing the corresponding 'count' field values of mi by appending additional nodes with 'count' field values set to 1 to the branch e_b.

For T_{r2}, reverse T_{r2} to get T_{r2r}.

Construct the tree for T_{r2r} as described earlier which is called the Suffix Tree.

Let TR1 be the root of Prefix Tree corresponding to T_{r1} and let TR2R be the root of the Suffix Tree corresponding to T_{r2r}.

2.2 Clustering Algorithm

In order to cluster the test pattern, the test pattern is also partitioned into 2-partitions using the same partitioning criteria as used for the training patterns. Let c be the number of classes, k, the number of nearest neighbours. The algorithm proceeds as follows.
For each branch b_l in the Prefix-Tree TR1.

Find the matches between the test pattern and the branch b_l. let it be C_1^l. Find k largest counts in decreasing order. Let them be $C_1^l, C_1^2, \ldots, C_1^k$ .Let the corresponding labels be $O_1^l, O_1^2, \ldots, O_1^k$.
Similarly, for each branch b_m in Suffix-Tree TR2R,

Find the matches between the test pattern and the branch b_m. let it be C_2^l. Find k largest counts in decreasing order. Let them be $C_2^l, C_2^2, \ldots, C_2^k$ and the corresponding labels be $O_2^l, O_2^2, \ldots, O_2^k$
For i= 1 to k

For j= 1 to k

Find $C_p = C_i + C_j$ if $O_i == O_j$ where $0 \leq p \leq k-1$.
Find k largest counts in decreasing order among all C_p where $0 \leq p \leq k$.
Compute the weight , $W_p = 1 - (C_k - C_p)/(C_p - C_1)$
For n = 1 to c

$Sum_n = \sum 1m[W_m]$ where $(O_m == n)$
Output (label = O_x) for which Sum_x is maximum for $x \in 1, 2, \ldots, c$

2.3 Comparision of the Prefix-Suffix Trees and the PC-Tree

The PC-tree[8] compacts the database by merging the nodes of the patterns having the same prefix. But in Prefix-Suffix Trees, still compaction is achieved by merging the branches of the trees 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 Fig.2A. The first column gives the transaction number, the second column gives the set of features and the last column gives the label. The partitioned set of transactions Part1 and Part2 are given in Fig.2B and Fig.2C and the set of transactions of Part2 in reverse order are given in Fig.2D respectively. The Prefix- tree , The Suffix-tree and the PC-tree for the set of transactions is given in Fig.3A, Fig.3B and Fig.3C 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.

3 Experiment and Results

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

Tr.No.	Features	Label
1	1,2,3,4,5,8,9,10,11,12,14,15,16	0
2	1,2,3,4,7,10,11,12,14,15,16	0
3	2,3,4,5,6,12,14,15,16	0
4	2,4,5,7,9,12,13,14	3
5	2,4,5,6,8,12,13,14	3

A. Sample set of transactions

Tr.No.	Features	Label
1	1,2,3,4,5,8	0
2	1,2,3,4,7	0
3	2,3,4,5,6	0
4	2,4,5,7	3
5	2,4,5,6,8	3

B.Set of Transactions in Part1

Tr.No.	Features	Label
1	9,10,11,12,14,15,16	0
2	10,11,12,14,15,16	0
3	12,14,15,16	0
4	12,13,14	3
5	12,13,14	3

C.Set of Transactions in Part2

Tr.No.	Features	Label
1	16,15,14,12,11,10,9	0
2	16,15,14,12,11,10	0
3	16,15,14,12	0
4	14,13,12	3
5	14,13,12	3

D.Set of Transactions of Part2 in reverse order

Fig. 2. Sample set of Transactions

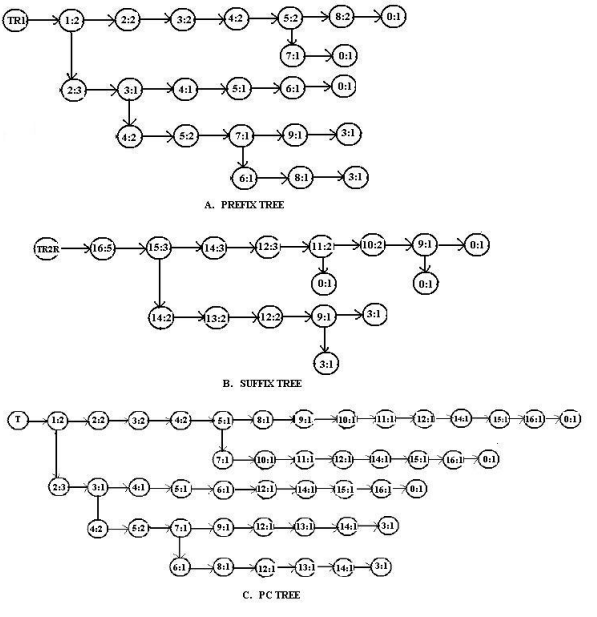

Fig. 3. Prefix Tree, Suffix Tree and PC Tree for the transactions in Fig.2

Table 1. Comparison of the Prefix-Suffix Trees, PP-structure, PC-Tree, PPC-Tree and K-NNC based algorithms

Expt-No	Dataset	Algorithm	Storage space in Bytes	Time in secs training+test time	Accuracy
1	OCR data (2000 patterns)	Prefix-Suffix Trees based alg.	1106880	120	91.45
		PP-structure based alg.	1123696	34	89.56
		PC-tree based alg.	1406528	153	89.56
		PPC-tree based alg. (with 4 parts)	1119280	131	41.58
		k-NNC	1544000	18	89.44
2	OCR data (4000 patterns)	Prefix-Suffix Trees based alg.	2046928	230	93.07
		PP-structure based alg.	2070112	110	92.04
		PC-tree based alg.	2717792	245	91.96
		PPC-tree based alg. (with 4 parts)	1970304	198	41.01
		k-NNC	3088000	31	91.9
3	OCR data (6670 patterns)	Prefix-Suffix Trees based alg.	3202576	315	94.12
		PP-structure based alg.	3216400	863	93.76
		PC-tree based alg.	4408256	386	93.61
		PPC-tree based alg. (with 4 parts)	3812320	314	64.3
		k-NNC	5149240	47	93.55
4	USPS data	Prefix-Suffix Trees based alg.	7877088	435	93.32
		PP-structure based alg.	7991504	554	93.27
		PC-tree based alg.	10030656	537	92.68
		PPC-tree based alg. (with 4 parts)	6490336	337	92.23
		k-NNC	7495148	39	93.27
5	MNIST data	Prefix-Suffix Trees based alg.	107430848	25317	96.7
		PP-structure based alg.	108528480	5996	96.5
		PC-tree based alg.	126535296	28451	96.5
		PPC-tree based alg. (with 4 parts)	77355312	17182	68.3
		k-NNC	188400000	cannot be stored in main mem.	cannot be implemented as it can't be stored

3.1 Dataset1: 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. Each class has approximately 670

training patterns and 333 test patterns. 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).

3.2 Dataset2: USPS Data

This is a dataset which is a collection of handwritten digits scanned from the U.S. postal services. There are 7291 patterns in the training set and 2007 patterns in the test set and 10 classes. Each pattern represents a digit and has 256 features.

3.3 Dataset3: MNIST Data

This is a data which is a mixture of the NIST(National Institute of standards and technology) special database 3 and 1. 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 has 784 features. We have compared our algorithm with the PC-tree, PPC-tree, the Prefix-postfix structure and the k-NNC algorithms for all the above datasets and the results are tabulated in Table 1. From the table, we observe that the Prefix-Suffix trees based algorithm consumes less space than the PC-tree, the Prefix-Postfix structure and the k-NNC algorithm without sacrificing for the classification accuracy. For all the datasets, dataset 1 , 2 and 3 we observe that the accuracy is increased by a certain order. All the experiments were executed on Xeon processor based Dell precision 670 workstation having a clock frequency of 3.2 GHZ and 1 GB RAM.

4 Conclusion

In this paper, a novel scheme called Prefix-Suffix Trees for compact storage of patterns is proposed which stores the transactions of a transaction database in a compact way. This scheme is complete, order independent and incremental. The use of this scheme in clustering is given and the effectiveness of the algorithm is established by comparing the scheme with the PC-tree, PPC-tree, Prefix-Postfix structure based algorithms and the benchmark algorithm k-NNC. The new scheme is found to be more compact than the PC-tree without sacrificing for the accuracy as shown. 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. Moore, A., Lee, M.S.: Cached Sufficient statistics for efficient machine learning with large datasets. Journal of Artificial Intelligence Research 8, 67–91 (1998)
2. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice-Hall, Englewood Cliffs (Advanced Reference Series)

3. Jain, A.K., Murty, M.N., Flynn, P.J.: Data Clustering: A Review. ACM Computing Surveys 31(3), 264–323 (1999)
4. Pujari, A.K.: Data Mining techniques. University Press, New Haven (2001)
5. Friedman, J.H., Bentley, J.L., Finkel, R.A.: An algorithm for finding best matches in logarithmic expected time. ACM trans. Math software 3(3), 209–226 (1997)
6. Viswanath, P., Murthy, M.N.: An incremental mining algorithm for compact realization of prototypes. Technical Report, IISC, Bangalore (2002)
7. Prakash, M., Murthy, M.N.: Growing subspace pattern recognition methods and their neural network models. IEEE trans. Neural Networks 8(1), 161–168 (1997)
8. Ananthanarayana, V.S., NarasimhaMurty, M., Subramanian, D.K.: Tree structure for efficient data mining using rough sets. Pattern Recognition Letters 24, 851–886 (2003)
9. http://www.cs.cmu.edu/15781/web/digits.html
10. http://wwwi6.informatik.rwthaachen.de/~keysers/usps.html
11. Duda, R.O., Hart, P.E.: Pattern Classification and Scene Analysis. Wiley, New York (1973)
12. Ravindra, T., Murthy, M.N.: Comparison of Genetic Algorithms based prototype selection scheme. Pattern Recognition 34, 523–525 (2001)
13. Pai, R.M., Ananthanarayana, V.S.: A novel data structure for efficient representation of large datasets in Data Mining. In: Proceedings of the 14th international Conference on Advanced Computing and Communications, pp. 547–552 (2006)