AN EXPERIMENTAL STUDY OF THE EFFECT OF FREQUENCY OF CO-OCCURRENCE OF FEATURES IN CLUSTERING

Radhika M. Pai[1] , Ananthanarayana V.S.[2]
[1]*Department of Computer Science &Engg., M.I.T., Manipal-576104, Karnataka, INDIA*
[2]*Department of Information Technology, N.I.T.K. Surathkal, P.O. Srinivasanagar – 575025, Karnataka, INDIA.*
E-mail : radhikampai@rediffmail.com, anvs@nitk.ac.in

ABSTRACT

In this paper, an attempt has been made to explore the effect of frequency of co-occurrence of features on the accuracy of the clustering results. This has been achieved by incorporating the frequency component in the clustering algorithm. The frequency, we mean here is the number of times the sequence of features appear in the data set. We try to utilize this component in the algorithm and study its effect on the resultant accuracy. The algorithm we have used is the PC(pattern count)-tree based clustering algorithm. The PC-tree is a compact and complete representation of the data set. It is data order independent and incremental. It can be applied to changing data and changing knowledge. i.e. dynamic databases. This algorithm is based on a compact data structure called PC-tree. The node of the PC-tree has, in addition to other fields a count field, which keeps track of the count of the number of features shared by the pattern. In the literature, the PC-tree was used for clustering and the count field was used only to retrieve back the transactions. In this paper, we try to make use of this field in clustering. We have also used the partitioned PC-tree based algorithm and studied the effect of frequency on the accuracy. We have conducted extensive experiments with the OCR handwritten digit dataset, a real dataset and observed the effect of frequency on the clustering results. The results of all our experiments are tabulated.

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]. 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 PC-tree[8] and the PPC-tree[6] based algorithms require only a single database scan and have been used for clustering of the handwritten digit dataset. In both these algorithms, the notion of frequency was not used. Most of the algorithms which use frequency is restricted to document clustering where the frequency of the occurrence of words is considered. The use of frequency for numeric data has not been exploited yet. Here we try to exploit the notion of frequency for the clustering of numeric dataset.

II. MOTIVATION FOR THE PRESENT WORK

Till now, work has been carried out on clustering, but the notion of frequency was till now restricted to the clustering of documents where the frequency of the occurrence of words was considered. Here we try to use the notion of frequency in clustering of numeral datasets. The reason why we have used the PC-tree is that the PC-tree already stores the frequency in the form of "count" value and we try to use this component in the algorithm so that there is no extra space consumed for storing the frequency.

III. OVERVIEW OF THE PC-TREE BASED ALGORITHM

The PC-tree[8] is a tree constructed from the data set. The node structure of the tree has four fields.

They are

'Feature' specifies the feature value of a pattern.

1-4244-0779-6/07/$20.00 ©2007 IEEE

'Count' specifies the number of patterns represented by a portion of the path reaching this node.

'Chld_ptr' represents the pointer to the following path.

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

Figure 1. shows the node format of a PC-tree.

Feature	Count	Chld_ptr	Sib_ptr

Figure 1. Node structure of the PC-tree

For completeness, the construction of the PC-tree and the clustering algorithm based on the PC-tree are given in Figure 2. and Figure 3. respectively.

Inputs:
let T_R be the database.
let root of the PC-tree be T.
For each pattern, $t_i \in T_R$
let m_i be the set of positions of non-zero values in t_i.
if no sub-pattern starting from T exists corresponding to m_i
then
create a new branch, with nodes having 'feature' fields as values of m_i and
'count' fields with values set to 1.
else
put the values of m_i in an existing branch, e_b by incrementing corresponding 'count' field values of the nodes in e_b. put the remaining values of m_i by appending additional nodes with 'count' field values set to 1 to the branch e_b.

Figure 2. Construction of the PC-tree

The PC-tree can be used as an abstract representation for the training patterns in supervised clustering. The test data patterns are used for testing . The clustering algorithm using PC-tree is given in Figure 2.

Generate PC-tree using T_R
For each $s_i \in T_S$
find n_i, set of positions of non-zero values corresponding to s_i.
for each branch $b_j \in$ PC-tree
Count the number of common features between n_i and b_j . let it be C_{ij}.
find k largest counts in descending order. let them be C_{i1}, C_{i2}, ...C_{ik}, and let the corresponding branches in the PC-tree be b^1, b^2, ... , b^k.
Compute the weight, $w_l = 1-(C_{il} - C_{ik}) / (C_{i1} - C_{ik})$, for each l=1,....k.
for n = 1 to number of classes do
$sum_n = \sum_{m=1}^{k}(w_m)$ where $(o_m == n)$

label = o_x, if sum_x is maximum, for x=1 to no of classes.
if (label == label of s_i) then
correct = correct + 1.
CA = (correct/ $|T_S|$ x 100) $|T_S|$ is the number of test patterns.

Figure 3. PC-tree based clustering algorithm

IV. THE PROPOSED METHOD

In the proposed method, the construction of the PC-tree remains the same. In the recognition phase, we match the incoming test data set with all the branches of the PC-tree and find the branch which matches more with the test data set by computing the distance and recording the minimum distance. We call this the nearest neighbour branch of the test set. In this way we compute the K-nearest neighbours and record the distance and the label. In addition to this, our proposed algorithm also records the frequency count of the nearest branches as the minimum of the count values of all the features in that branch. This frequency is used in computing the weightages. In the first method, we multiply the formula for computing the weightage by the frequency and take it as the weightage for that entry.

In the second method, we repeat each entry in the k-nearest neighbour list as many times as the frequency and take the resultant k-nearest neighbours.

The resultant algorithms are given in Figure 4. and Figure 5. respectively.

Generate PC-tree using T_R
For each $s_i \in T_S$
find n_i, set of positions of non-zero values corresponding to s_i.
for each branch $b_j \in$ PC-tree
Count the number of common features between n_i and b_j . let it be C_{ij}.
find k largest counts in descending order. let them be C_{i1}, C_{i2}, ...C_{ik}, and let the corresponding branches in the PC-tree be b^1, b^2, ... , b^k and let the frequency of the corresponding branch be $lcount_1$, $lcount_2$, $lcount_k$.
Compute the weight, $w_l = (1- (C_{il} - C_{ik}) / (C_{i1} - C_{ik})) * lcount_i$, for each l=1,....k.

for n = 1 to number of classes do
$sum_n = \sum_{m=1}^{k}(w_m)$ where $(o_m == n)$

label = o_x, if sum_x is maximum, for x=1 to no of classes.

if (label == label of s_i) then

correct = correct + 1.

CA = (correct/ $|T_S|$ x 100) $|T_S|$ is the number of test patterns.

Figure 4. Proposed method 1

Generate PC-tree using T_R

For each s_i € T_S

find n_i, set of positions of non-zero values corresponding to s_i.

for each branch b_j € PC-tree

Count the number of common features between n_i and b_j . let it be C_{ij}.

find k largest counts in descending order. let them be C_{i1}, C_{i2}, ...C_{ik}, and let the corresponding branches in the PC-tree be b^1, b^2, ... , b^k and let the frequency of the corresponding branch be $lcount_1$, $lcount_2$, $lcount_k$.

Repeat the each of the k largest counts , depending on the lcount value of each entry.

Takee only the k-nearest neighbours , after repeating the entries.

Compute the weight, $w_l = 1- (C_{il} – C_{ik}) / (C_{i1} – C_{ik})$, for each l=1,....k.

for n = 1 to number of classes do

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

label = o_x, if sum_x is maximum, for x=1 to no of classes.

if (label == label of s_i) then

correct = correct + 1.

CA = (correct/ $|T_S|$ x 100) $|T_S|$ is the number of test patterns.

Figure 5. Proposed method 2

V. EXPERIMENTAL RESULTS

We have conducted the experiments on the OCR handwritten digit dataset. There are 6670 patterns in the training set, 3333 patterns in the test set and 10 classes.[6,7,8]. 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. The results of the proposed method 1 and proposed method 2 is shown in Table 1.

Further experiments were conducted by generating some synthetic training set and observing the effect of frequency on the accuracy. All the results of the experiments are tabulated in Table 1.

We also have conducted the experiments by using the PPC-tree based clustering algorithm[6]. The PPC-tree based clustering algorithm is similar to PC-tree based algorithm, but it partitions the dataset vertically and constructs the PC-tree separately for each partition. The test pattern is likewise partitioned and matched with each partition independently. those results are also tabulated in Table-1.

Other experiments were conducted by giving the training set as input more than once and multiplying the entries in the nearest neighbour list based on this value. All the results of the experiments are tabulated.

All the above mentioned experiments were conducted on an Intel Pentium 4 machine with 248 MB RAM and 2.4 GHz clock frequency running LINUX .

VI. CONCLUSION

In this paper, a detailed experimental study is conducted to observe the effect of frequency of the occurrence of features on the accuracy of clustering results. The experiments have been conducted on the handwritten digit dataset and two different algorithms were used by incorporating the frequency component in the algorithm. The results of all the experiments are tabulated. Several variants of the algorithms were tried by generating the synthetic set, duplicating the entries in NN-list, by trying with different number for K neighbours and so on. The results tabulated are for the values of K for which the Accuracy is maximum. Though the results are slightly lesser in accuracy than mentioned in literature, nevertheless, the experiments still highlight that the frequency component plays a vital role in clustering though not for this dataset as the frequency in most of the cases was 1 except for a particular label for which the frequency was above 20.

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

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

Table 1. Results of all the Experiments

Clustering Algorithm Used	Modified algorithm	% Accuracy
PC-tree based Clustering Alg.	Proposed method 1	93.39
	Proposed method 2	91.89
	Proposed method1 with synthetic set1	93.58
	Proposed method1 with synthetic set2	93.03
	Duplicating the training set and duplicating the NN based on freq.	93.12
	Duplicating the training set and duplicating each of the entry in NN	93.09
	triplicating the training set and triplicating each of the entry in NN	93.64
	triplicating the training set and triplicating each of the entry in NN with synthetic set1	93.76
	triplicating the training set and triplicating each of the entry in NN with synthetic set2	92.97
Partitioned PC-tree based Clustering Alg.	Proposed method 1	94.72
	triplicating the training set and triplicating each of the entry in NN	94.78
	triplicating the training set and triplicating each of the entry in NN	94.54