An efficient incremental protein sequence clustering algorithm

P. A. Vijaya, M. Narasimha Murty and D. K. Subramanian
Department of Computer Science and Automation
Indian Institute of Science
Bangalore - 560012, India
080-293-2368
{pav,mnm,dks}@csa.iisc.ernet.in

Abstract— Clustering is the division of data into groups of similar objects. The main objective of this unsupervised learning technique is to find a natural grouping or meaningful partition by using a distance or similarity function. Clustering is mainly used for dimensionality reduction, prototype selection/abstractions for pattern classification, data reorganization and indexing and for detecting outliers and noisy patterns. Clustering techniques are applied in pattern classification schemes, bioinformatics, data mining, web mining, biometrics, document processing, remote sensed data analysis, biomedical data analysis, etc., in which the data size is very large. In this paper, an efficient incremental clustering algorithm - 'Leaders-Subleaders' - an extension of leader algorithm, suitable for protein sequences of bioinformatics is proposed for effective clustering and prototype selection for pattern classification. It is another simple and efficient technique to generate a hierarchical structure for finding the subgroups/subclusters within each cluster which may be used to find the superfamily, family and subfamily relationships of protein sequences. The experimental results (classification accuracy using the prototypes obtained and the computation time) of the proposed algorithm are compared with that of leader based and nearest neighbour classifier (NNC) methods. It is found to be computationally efficient when compared to NNC. Classification accuracy obtained using the representatives generated by the Leaders-Subleaders method is found to be better than that of using leaders as representatives and it approaches to that of NNC if sequential search is used on the sequences from the selected subcluster.

1. INTRODUCTION

Clustering is an active research topic in pattern recognition, data mining, statistics and machine learning with diverse emphasis. The earlier approaches do not adequately consider the fact that the data set can be too large and may not fit in the main memory of some computers. In bioinformatics, the number of protein sequences is now more than half a million. It is necessary to devise efficient algorithms to minimize the disk I/O operations. The problem we have considered here is : Given a set of protein sequences, design and implement efficient clustering techniques to find meaningful partitions/groupings so as to improve the classification accuracy and reduce the disk I/O operations, computation time and space requirements. Also, to find an alternative and efficient scheme to generate a hierarchical structure of protein sequences.

This paper is organized as follows. In section 2, various clustering approaches used so far are mentioned in brief. Section 3 contains basics of molecular biology, significance of protein sequence comparison and clustering and related work. Section 4 contains the details of the proposed method. Experimental results and discussions are presented in section 5. Conclusions and further research scope are provided in section 6.

2. CLUSTERING TECHNIQUES

Clustering techniques are classified into hierarchical and partitional methods. Hierarchical clustering algorithms can be either divisive (top-down) or agglomerative (bottom-up) [1], [2], [3]. Single link and complete link are hierarchical agglomerative clustering algorithms. For both of these, similarity matrix requires $O(n^2)$ space and time complexity is $O(n^2d)$ for distance computation and $O(n^3d)$ for complete clustering procedure, where d is the dimensionality. Therefore, they are not suitable for large data sets.

K-means, K-medoids and K-modes are some of the examples for partitional clustering approaches [1], [2], [3]. K-means and K-medoids are based on K centroids and medoids of the initial partitions respectively and are iteratively improved. K-means has a time complexity of $O(dtK$), where t is the number of iterations and space complexity of $O(Kd)$ and centroid can be defined only for numerical data. K-modes is an extension of K-means for categorical data. PAM (Partitioning Around Medoids) algorithm [3] selects K patterns arbitrarily as medoids and then iteratively improves upon this selection. Its time complexity is $O(K(n-K)^2)$ and space complexity is $O(Kd)$. For large data sets even conventional K-means, K-medoids, PAM and K-modes involve lot of I/O operations and computations and hence are not suitable. The clustering approaches designed for large data sets are described in the following section.

Clustering of large data sets

In data mining applications, both the number of patterns and features are typically large. In bioinformatics, DNA/protein sequences are very long and the sequences are of unequal lengths. Data cannot be stored in main memory and have to

0-7803-7651-X/03/$17.00 ©2003 IEEE

be transfered from secondary storage as and when required. Single link, complete link, K-means and K-medoids based algorithms are not feasible for large data sets. Following are some of the clustering approaches that have been used for large data sets.

CLARA and CLARANS [3] are improved versions of PAM designed for large data sets. But they are computationally expensive because of the large number of iterations carried out to get good set of medoids. DBSCAN [3] is a partitional clustering technique and is less sensitive to outliers and can discover clusters of irregular shapes. Its time complexity is $O(nlog(n))$ and space complexity is $O(n)$. CURE [3] is a hierarchical agglomerative clustering scheme designed to find clusters of arbitrary shapes and sizes and is robust to outliers. Its time and space complexity is $O(n^2)$. There are other clustering approaches such as grid based clustering, subspace clustering, self organizing map, clustering with frequent item sets and so on. But incremental clustering methods such as leader [17] and BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) [3] are more efficient for large data sets as they involve few database scans (less I/O operations). Leader is a simple partitional clustering technique suitable for any type of data set whereas BIRCH is an incremental hierarchical agglomerative clustering technique and is suitable only for numerical data sets.

Protein sequence clustering method and the related work are discussed in the following section.

3. PROTEIN SEQUENCE CLUSTERING

Basics of molecular biology

DNA (Deoxyribose Nucleic Acid) is a sequence of 4 bases/nucleotides Adenine(A), Guanine(G), Thymine(T) and Cytosine(C). DNA is transcribed into RNA (Ribose Nucleic Acid) which contains Uracil(U) instead of Thymine. RNA (mRNA) is further translated into proteins. Proteins are sequences composed of an alphabet of 20 amino acids. Three bases/nucleotides of a DNA/mRNA code for an amino acid. In protein sequences, the amino acids are abbreviated using single letter codes such as A for Alanine, S for Serine and so on [5], [6]. The linear sequence of amino acids in a polypeptide chain is known as the primary structure of a protein. Alpha helices and beta sheets are the secondary structures of a protein.

Significance of protein sequence comparison and clustering

DNA/protein sequence may change after few generations because of the three edit operations - insertion, deletion and substitution. Pairwise sequence alignment is used to compare and cluster sequences. There are two types of pairwise sequence alignments, local and global [5], [6]. Local alignment helps in finding conserved amino acid patterns in protein sequences. Local alignment programs are based on Smith Waterman algorithm [7]. In global alignment attempts are made to align the entire sequence using as many characters as possible, up to both ends of each sequence. Global alignment programs are based on Needleman and Wunsch algorithm [8]. For finding the similarity between two protein sequences, PAM250 or BLOSUM60 or BLOSUM62 scoring matrix is used [5], [6]. They contain the substitution values/costs for all pairs of 20 amino acids. Gap penalty is to be properly selected. Higher the score value, sequences are more similar.

Score of an alignment

Let sequence $a = ATTGGTA$ and sequence $b = AGGCTC$. One possible alignment is

$a^0 = ATTGG-TA$
$b^0 = A--GGCTC$

Score of an alignment = $W(a^0, b^0) = \sum_i D(a_i^0, b_i^0)$

where, $1 \leq i \leq l$, l is the length of the aligned sequence and D is the cost of an operation. The costs are defined by biologists for insertion, deletion and substitutions (for both match and mismatch). Scores are calculated for the subsequences aligned in local alignment and for the entire aligned length in case of global alignment. Protein sequence similarity score can be used in clustering similar sequences or classifying a new/test sequence to a known protein class/group/family.

Similar protein sequences, probably have similar biochemical function and three dimensional structure. If two sequences from different organisms are similar, they may have a common ancestor sequence and sequences are said to be homologous. Protein sequence clustering helps in classifying a new sequence, retrieve a set of similar sequences for a given query sequence, predicting the protein structure of an unknown sequence and finding the superfamily, family and subfamily relationships of protein sequences.

Related work

ProtoMap [10] is designed based on weighted directed graph and CluSTr [11] uses single link method. CLICK [12] uses graph theoretic and statistical techniques and SCOP [14] is based on hierarchical clustering. Krause [9] has used set-theoritic method and single linkage clustering for constructing the phylogeny tree of protein sequences. Eui Hong et al. [13] have designed a hypergraph based model using frequent item sets for clustering data. Guralnik et al. [15] have designed a K-means based algorithm for clustering protein sequences. Eva Bolten et al. [16] have used transitive homology, a graph theoritic based approach for clustering protein sequences for structure prediction.

Single linkage clustering method is computationally very expensive for large set of protein sequences and it also suffers from chaining effect. Even in graph based approaches the distance matrix values are to be calculated and is also expensive for large data sets. In both the cases, distance matrix may not be accomodated in main memory and it increases the

disk I/O operations. Here, we propose a method to extend the leader algorithm as it is a simple incremental clustering algorithm with a time complexity of $O(n)$ (one database scan) and space complexity of $O(Ld)$. In the proposed method, the leader algorithm is extended to generate a hierarchical structure with clusters (leaders as representatives) in the first level and subclusters (subleaders as representatives) in the second level. The proposed method and the experimental results are discussed in the following sections.

4. Proposed Method

Leader is an incremental algorithm in which each of the L clusters is represented by a leader. L clusters are generated using a suitable threshold value. In this method, the first pattern is selected as the leader of a cluster and the remaining patterns are classified depending on the existing leaders (Lds) or may become leader of a new cluster. As an extension of leader algorithm, we have implemented Leaders-Subleaders algorithm. In this method, after finding L leaders using the leader algorithm, subleaders (Sublds) are generated within each cluster represented by a leader, choosing a suitable subthreshold value. Thus, Leaders-Subleaders algorithm creates L clusters/leaders and SL_i subleaders in the i^{th} cluster as shown in Figure 1. Subleaders are the representatives of the subclusters and they inturn help in classifying the given new/test pattern more accurately. This algorithm generates subgroups/subclusters within each cluster as required in many applications. Thus, it generates a hierarchical structure and this procedure may be extended to more than two levels. A two level hierarchical structure as shown in Figure 2 can be generated in only two database scans and is computationally less expensive compared to other hierarchical clustering algorithms. Space complexity of Leaders-Subleaders algorithm is $O((L+SL)d)$, where L and SL are the total number of leaders and subleaders respectively and their sum is less than the total number of patterns - n.

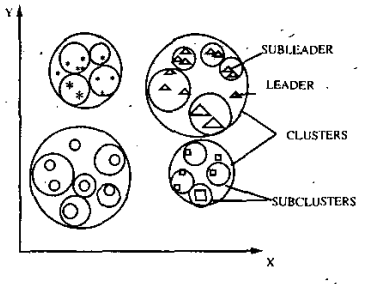

Figure 1. Clusters in Leaders-Subleaders algorithm

The algorithms for leader and Leaders-Subleaders are given below.

Leader algorithm :

Training:

1. Select threshold value

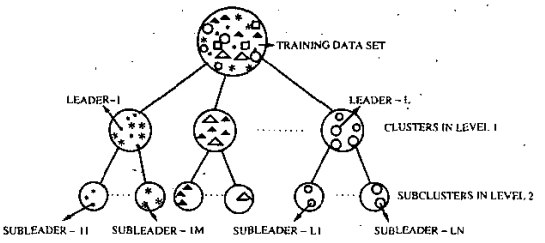

Figure 2. Hierarchical structure in Leaders-Subleaders algorithm

2. Initialize a leader and add it to leader list and set leader counter, L = 1
3. For all patterns, i = 2 to n
{
Calculate the similarity score with all leaders
Find the nearest leader
If (similarity score with nearest leader
> threshold)
{
Assign it to the nearest leader
Mark the cluster number
Add it to member list of this cluster
Increment member count of this cluster
}
else
{
Add it to leader list
Increment leader counter, L = L + 1
}
}

Testing :

1. Initialise the counter, total-match = 0
2. Do for k = 1 to m test patterns
{
a. Calculate the similarity score with all leaders
b. Find the nearest leader - nl
c. predicted class = class of nl
d. If (predicted class == actual class)
total-match = total-match + 1
}
*3. Accuracy = (total-match / total test patterns) * 100*

Leaders-Subleaders algorithm :

Training:

1. Training part of the Leader algorithm to generate L clusters/leaders
2. Select subthreshold value (> threshold value)
3. Do for i = 1 to L clusters
{
a. Initialize a subleader, add it to subleader list and set counter, $SL_i = 1$
b. For j = 2 to member count of i^{th} cluster

```
    {
    Calculate the similarity score with all subleaders
    Find the nearest subleader
    If (similarity score with nearest subleader
        > subthreshold)
      {
      Assign it to the nearest subleader
      Mark the subcluster number
      Add it to member list of this subcluster
      Increment member count of this subcluster
      }
    else
      {
      Add it to subleader list
      Increment subleader counter, SL_i = SL_i + 1
      }
    }
  }
4. Initialize counter SL = 0, For i = 1 to L {SL = SL + SL_i}
```

Testing:

```
1. Initialise the counter, total-match = 0
2. Do for k = 1 to m test patterns
  {
    a. Calculate the similarity score with all leaders
    b. Find the nearest leader - nl
    c. Calculate the similarity score with all subleaders
       under that leader
    d. Find the nearest subleader - ns
    e. If (similarity score from nl >
            similarity score from ns)
         predicted class = class of nl
       else predicted class = class of ns
    f. If (predicted class == actual class)
         total-match = total-match + 1
  }
3. Accuracy = (total-match / total test patterns) * 100
```

5. EXPERIMENTAL RESULTS

To evaluate the performance of leader and Leaders-Subleaders algorithm, a protein sequence family with known subfamilies/groups is considered.

Protein sequence data set:

Protein sequences of HLA protein family have been collected from www.obi.ac.uk/imgt/hla. It contains 1609 sequences grouped into 19 classes. Protein sequences of AAA protein family have been collected from http://aaa-proteins.uni-graz.at/AAA/AAA-Sequences.text. AAA protein family sequences have been categorized into 6 classes according to their functions and 227 sequences have been considered from this family. From Globins protein family, sequences have been collected from 4 different classes and 629 sequences have been selected from the data set provided along with the software package hmmer-2.2g (ftp://ftp.genetics.wustl.edu/pub/eddy/hmmer/). Thus, totally we have considered 29 different classes containing the sequences according to protein functions. We have considered these groups of protein sequences as they have been classified according to functions by scientists/experts. The data set considered has totally 2565 sequences. From this, randomly 1919 sequences were selected for training and 646 for testing. The local alignment program provided by Xiaoqui et al. [18] is used with necessary modifications for finding the similarity score. PAM250 scoring matrix is used in our program.

The experiments were done on Intel pentium-4 processor based machine having a clock frequency of 1700 Mhz and 512 MB RAM. In each case, different threshold and subthreshold values were used for leader and Leaders-Subleaders algorithms. The best results are reported here due to space constraints. From the results obtained (as shown in Table 1), it is evident that both the algorithms performed well compared to the nearest neighbour approach [4] in terms of computation time. Leaders-Subleaders algorithm gives better accuracy compared to leader algorithm. Classification accuracy increases when sequential search is used for the selected cluster (Seq-Ser-Lds) or subcluster (Seq-Ser-Lds-Sublds). Training time is the time taken for generating leaders or leaders and subleaders. Testing time is the time taken for classifying all the test patterns. After the training phase, testing time is very less for leader and Leaders-Subleaders algorithms but increases for Seq-Ser-Lds and Seq-Ser-Lds-Sublds algorithms. Even then, they are less compared to NNC. Seq-Ser-Lds algorithm require more testing time as it searches an entire cluster and its accuracy can be as that of NNC. The leader and Leaders-Subleaders algorithms require only one and two database scans respectively. Time complexity of these algorithms is $O(n)$ as compared to single link algorithm whose time complexity is $O(n^3d)$.

Though the results are given here for a small data set, the proposed algorithm is applicable for large data sets. The results presented are for the case where data could be accomodated in the main memory. For large data sets, the prototypes selected are written to disk after the training phase. Then, only the time taken for the testing phase using these representatives is to be compared between the algorithms used. The space requirement will be reduced as only these representatives are to be stored in the main memory during the testing phase. Even if more number of prototypes are generated, search space is less as only part of the hierarchical structure is searched. The disk I/O time, which is more critical than the computation time, can be reduced for leader and Leaders-Subleaders algorithms as compared to single link algorithm.

6. CONCLUSIONS

In this paper, experimental results of Leaders-Subleaders algorithm on a protein sequence data set show that it performs well by properly tuning the threshold and subthreshold values. Sequential search in the selected cluster or subcluster gives classification accuracy nearly same as nearest neighbour method. Hierarchical structure with required number

Table 1. Comparison of time and classification accuracy

Algorithm	Thres-hold	Subthr-eshold	Lds	Sublds	Training time (secs)	Testing time (secs)	Classification Accuracy (%)
Nearest Neighbour	-	-	-	-	-	19104.02	99.84
Leader	350	-	68	-	653.53	601.05	56.96
	400	-	109	-	967.39	853.13	57.12
Seq-Ser-Lds	350	-	68	-	653.53	5933.94	99.84
	400	-	109	-	967.39	6053.14	99.84
Leaders-Subleaders	350	750	68	835	1303.91	1068.08	93.34
	400	800	109	816	1663.26	1321.95	94.27
Seq-Ser-Lds-Sublds	350	750	68	835	1303.91	2720.98	98.76
	400	800	109	816	1663.26	2776.19	98.91

of levels can be generated by using Leaders-Subleaders algorithm to find the subgroups/subclusters within each cluster at low computation cost and this may be used to find the superfamily, family and subfamily relationships in protein sequences. We further aim at clustering a large set of protein sequences consisting of sequences from various families and evaluate the performance of the proposed algorithm with necessary modifications. Also, we would like to compare the classification accuracy and computation time for large data sets with few more algorithms. The proposed algorithm can also be used on numerical data sets, text and web document collection.

REFERENCES

[1] A.K. Jain, M.N. Murty, and P.J. Flynn, "Data clustering : A Review," *ACM Computing Surveys*, Vol. 31, **3**, pp. 264–323, 1999.

[2] Pavel Berkhin, "Survey of Clustering Data Mining Techniques," *Technical Report (http://citeseer.nj.nec.com/berkhin02survey.html)*, 2002.

[3] Arun K Pujari, *Data Mining Techniques*, Universities Press (India) Private Limited, 2000.

[4] R. Duda, P. Hart, and D. Stork, *Pattern classification*, John Wiley, 2^{nd} edition, 2002.

[5] Peter Clote, and Rolf Backofen, *Computational Molecular Biology - An Introduction*, John Wiley & Sons, Ltd., August 2000.

[6] David W Mount, *Bioinformatics - Sequence and Genome Analysis*, Cold Spring Harbor Laboratory Press, New York, May 2002.

[7] T.F. Smith, and M.S. Waterman, "Identification of common molecular subsequences," *J. of Mol. Biology*, **147**, pp. 195–197, 1981a.

[8] S.B. Needleman, and C.D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of the proteins," *J. of Mol. Biology*, **48**, pp. 443–453, 1970.

[9] A. Krause, "Large scale clustering of protein sequences," *Ph.D. Dissertation*, Berlin, 2002.

[10] G. Yona, N. Linial, and M. Linial, "ProtoMap: automatic classification of protein sequences and hierarchy of protein families," *Nucleic Acids Research*, **28**, 2000.

[11] E.V. Kriventseva, W. Fleischmann, E.M. Zdobnov, and G. Apweiler, "CluSTr: a database of clusters of SWISS-PROT+TrEMBL proteins," *Nucleic Acids Research*, **29**, 2001.

[12] R. Sharan, and R. Shamir, "CLICK: A clustering algorithm with applications to gene expression analysis," *Proc. of* 8^{th} *ISMB*, 2000.

[13] Eui Hong, G. Karypis, V. Kumar, and B. Mobasher, "Clustering in a high dimensional space using hypergraph models," *Research issues on Data Mining and Knowledge Discovery*, 1997.

[14] L.L. Conte, B. Ailey, T.J.P. Hubbard, S.E. Brenner, A.G. Murzin, and C. Chotia, "SCOP: a structural classification of Protein database," *Nucleic Acids Research*, **28**, 2000.

[15] V. Guralnik, and G. Karypis, "A scalable algorithm for clustering sequential data," *Proc. of* I^{st} *IEEE conference on Data Mining*, 2001.

[16] Eva Bolten, Alexander Schliep, and Sebastian Schneckener, "Clustering Protein Sequences-Structure prediction by transitive homology," *GCB*, 1999.

[17] Helmuth Spath, *Cluster analysis algorithms*, Ellis Horwood, Chichester, UK, 1980.

[18] Xiaoqui Huang, and Webb Miller, "A Time-Efficient, Linear-Space Local Similarity Algorithm," *Advances in Applied Mathematics*, **12**, pp. 337–357, 1991.