International Journal of Computer Science & Applications
Vol. I, No. II, pp. 61 - 75

An Efficient Hierarchical Clustering Algorithm for Protein Sequences

P.A. Vijaya, M. Narasimha Murty and D.K. Subramanian
Department of Computer Science and Automation
Indian Institute of Science
Bangalore - 560012, India.
{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 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. Even if more number of prototypes are generated, classification time is less as only a part of the hierarchical structure is searched in Leaders-Subleaders method.

Keywords: Clustering; prototypes; classification; protein sequences; incremental; leaders; subleaders; hierarchical structure.

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. It is necessary to investigate the principle of clustering to devise efficient algorithms to minimize the I/O operations and space requirements and to get appropriate prototypes/abstractions to increase the classification accuracy. One such application where efficient clustering techniques are required is bioinformatics. Some details about molecular biology has been included in the next subsection for better understanding.

1.1 Basics of molecular biology

Cells are the fundamental working units of every living system. All the instructions needed to direct the cell activities are contained within the chemical DNA (Deoxyribose Nucleic Acid). DNA from all organisms is made up of the same chemical and physical components. The GENOME is an organism's complete set of DNA and they vary widely in size. DNA in the human genome is arranged into 24 distinct chromosomes.

Each chromosome contains many genes, the basic physical and functional units of heredity. Genes are specific sequences of bases that encode instructions on how to make proteins. Although genes get a lot of attention, it's the proteins that perform most life functions and even make up the majority of cellular structures. The constellation of all proteins in a cell is called its proteome. Studies to explore protein structures and their functions is known as proteomics.

DNA is made up of deoxyribose sugar, phosphate, and bases/nucleotides. Bases are Purines (Adenine(A) and Guanine(G)) and Pyrimidines (Thymine(T) and Cytosine(C)). A DNA molecule consists of two strands which are coiled around each other in a double helix. One DNA strand can be up to several hundred million nucleotides in length. DNA is a sequence of 4 bases/nucleotides abbreviated as A, G, T and C. DNA is transcribed into RNA (Ribose Nucleic Acid) which contains Uracil(U) instead of Thymine. RNA is single stranded. The messenger RNA (mRNA) serves as an intermediate between DNA and protein. Parts of the DNA are transcribed into mRNA and is further translated into proteins. The regions of genomic DNA that encode proteins are known as exons/coding regions. The regions of genomic DNA that do not encode proteins are known as introns/noncoding regions. 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 [1, 2]. Figure 1 shows the information flow in biology. It is the central dogma of molecular genetics [1]. 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. DNA/protein sequence may change after few generations because of the three edit operations - insertion, deletion and substitution. Similarity score between a pair of protein sequence can be obtained from sequence alignment procedures. Protein sequence similarity score can be used in clustering similar sequences or classifying a new/test sequence to a known protein class/group/family. Protein sequence alignment procedure and similarity score are discussed in the following section.

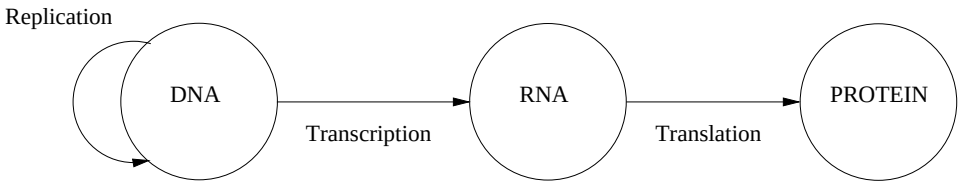

Figure 1: Information flow in biology

1.2 Protein sequence alignment and similarity score

Two sequences can be aligned by writing them across a page in two rows. Identical or similar characters are placed in the same column, and nonidentical characters can either be placed in the same column as a mismatch or opposite a gap in the other column. Sequences that can be readily aligned in this manner are said to be similar. Similar sequences, probably have the same function, regulatory role in the case of similar DNA molecules, or a similar biochemical function and three dimensional structure in the case of proteins. If two sequences from different organisms are similar, they may have a common ancestor sequence and sequences are said to be homologous. The alignment indicates the changes that could have occured between the two homologous sequences and a common ancestor sequence during evolution. 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 [1, 2]. Local alignment helps in finding conserved amino acid patterns in protein sequences. Local alignment programs are based on Smith Waterman algorithm [3]. 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 [4].

1.2.1 Score of an alignment

A sequence might change after few generations. The 3 edit operations are substitution, insertion and deletion.

Let

$$\sum = Alphabet, \sum = \{A, C, G, T\}$$

Edit operation is a pair

$$(x, y)\epsilon(\sum \bigcup \{-\}) \times (\sum \bigcup \{-\})$$

Let sequence/subsequence $a = ACCGGTA$ and sequence/subsequence $b = AGGCTG$. One possible alignment is

$$a^0 = ACCGG - TA$$

$$b^0 = A - -GGCTG$$

The sequence of edit operation S, is
$S = (C, -)(C, -)(-, C)(A, G)$ (i.e. Deletion, Deletion, Insertion and Substitution)

An alignment is a pair (a^0, b^0) where $a^0, b^0 \epsilon (\sum \bigcup \{-\})$ such that

$|a^0| = |b^0|$

There is no position i such that

$a_i^0 = - = b_i^o$

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

where, $1 \leq i \leq l_a$, l_a is the length of the aligned sequence and E is the cost/score of an operation. The costs/scores are defined by biologists for insertion, deletion, substitutions (for both match and mismatch), gap initiation/open and gap extension. If linear gap penalty is considered then a constant gap penalty is used for every insertion or deletion operation. In the case of affine gap penalty, the function used is $g(f) = gap\_open\_penalty + k^{'} gap\_extension\_penalty$, where $k^{'}$ is the contiguous number of gaps after a gap initiation/open. Dynamic programming techniques [1, 2] are used for finding the score from an optimal alignment and they cannot be discussed in detail here for space constraints.

For finding the similarity between two protein sequences, PAM250 or BLOSUM60 or BLOSUM62 scoring matrix is used. These matrices contain the substitution values/costs for all pairs of 20 amino acids. The gap penalties are to be properly selected and are suggested by biologists. Higher the score value, sequences are more similar. The scores for the different substitutions are stored in a substitution score matrix, which contains for every pair (A, B) of amino acids an entry S_{AB} (the score for aligning A with B). There are different manners in which a substitution matrix can be derived. First approach is that a biologist can always set up a score matrix that produces good alignments. The second approach is to derive a score matrix from physical/chemical properties. The third and most often used approach is a statistical one. PAM (Point Accepted Mutation) matrix was generated by Dayhoff [1, 2] using the statistical approach. PAM250 is the most widely used matrix. PAM matrix values were derived from closely related proteins. Protein sequence similarity score can be used in clustering similar sequences or classifying a new/test sequence to a known protein class/group/family.

1.3 Significance of protein sequence clustering

Molecular biology has undergone an incredibly rapid development, currently yielding so much raw data that efficient computer algorithms are mandatory for data analysis. The number of unique entries in all

protein sequence databases together exceeds now more than half a million. However, biological evolution lets proteins fall into so-called families, thus imposing a natural grouping. A protein family contains sequences that are evolutionarily related and/or share a common three dimensional fold. Similar protein sequences, probably have similar biochemical function and three dimensional structure. 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 family and subfamily relationships of protein sequences. Protein structure prediction helps in drug design for treatment of diseases. Thus bioinformatics has its great impact on society and medicine. Therefore, one aims at organizing the set of all proteins into clusters based on their sequence similarity. Various clustering techniques available are explained below in brief.

1.4 Clustering Techniques

Clustering techniques are classified into hierarchical and partitional methods.

Hierarchical clustering algorithms can be either divisive (top-down) or agglomerative (bottom-up) [5, 6, 7]. Divisive (top-down/splitting) methods start with the entire data in one cluster and form the hierarchy by splitting the data set into smaller blocks successively. Process is stopped depending on the number of clusters to be generated. Agglomerative clustering (bottom-up/merging) procedures start with - n singleton clusters and form the hierarchy by successively merging the clusters until desired number of clusters is obtained. Single link and complete link algorithms are of this type. In Single link method, the distance between two clusters C_1 and C_2 is the minimum of the distances $dist(X,Y)$, where $X \in$C$_1$ and $Y \in$C$_2$ (for dissimilarity index like Euclidean distance). It suffers from chaining effect. In Complete link method, the distance between two clusters C_1 and C_2 is the maximum of the distances $dist(X,Y)$, where $X \in$C$_1$ and $Y \in$C$_2$ (for dissimilarity index like Euclidean distance). It produces tightly bound or compact clusters. For single link and complete link algorithms, similarity matrix requires $O(n^2)$ space and time complexity is $O(n^2 d)$ for distance computation and $O(n^3 d)$ for complete clustering procedure, where n is the total number of patterns and d is the dimensionality. Therefore, they are not suitable for large data sets.

K-means and K-medoids are partitional clustering approaches [5, 6, 7]. 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 $O(nKdt)$, where t is the number of iterations and space complexity of $O(Kd)$ and centroid can be defined only for numerical data. PAM (Partitioning Around Medoids) algorithm [7] 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 K-means and PAM involve lot of I/O operations and computations and hence are not suitable.

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 sometimes and may have to 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 [7] are improved versions of PAM designed for clustering large data sets. But they are computationally expensive because of the large number of iterations carried out to get a good set of medoids. DBSCAN [7] is a partitional clustering technique which uses an R^* tree and is less sensitive to outliers and can discover clusters of irregular shapes. Its time complexity is $O(nlog(n)d)$ and space complexity is $O(nd)$. CURE [7] is a hierarchical agglomerative clustering scheme designed to find clusters for small, numerical, spatial data. Its time complexity is $O(N^2_{sample} d)$ and space complexity is $O(N^2_{sample})$ for small size, where N_{sample} is the number of samples used. Incremental clustering methods such as leader [8] and BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) [7] are more efficient for clustering large data sets as they involve a few database scans (less I/O operations). Leader is a simple partitional clustering technique suitable for both numerical and sequence data sets whereas BIRCH is an incremental hierarchical agglomerative clustering technique and is suitable only for numerical data sets.

The problem we have considered here is : Given a set of protein sequences, design and implement efficient clustering techniques to find a good set of prototypes from 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. Section 2 discusses the related work in protein sequence clustering. Section 3 contains the details of the proposed method. Experimental results and discussions are presented in section 4. Conclusions and further research scope are provided in section 5.

2 Related work in protein sequence clustering

A survey of protein sequence clustering approaches used so far is given below.

ProtoMap, designed by Yona et al. [9] offers a classification of all the sequences in the Swiss-Prot and TrEMBL database into groups of related proteins. Several common measures of similarity between protein sequences (Smith-waterman, FASTA, BLAST) are combined with two different scoring matrices (BLOSUM50 and BLOSUM62) to create an exhaustive list of neighboring sequences for each sequence in the database. These lists induce a representation of the protein space as a weighted directed graph whose nodes are the sequences. The weight of an edge connecting two sequences represents their degree of similarity. Clusters of related proteins correspond to strongly connected components of this graph. The analysis starts from a very conservative classification, based on highly significant similarities, that consists of many classes. Subsequently, classes are merged to account for less significant similarities. The process is repeated at varying confidence levels, where at each step the algorithm is applied on the classes of the previous classification, to obtain the next one, at the more permissive threshold. Consequently, a hierarchical organization of all proteins is obtained.

Clustr(Clusters of Swiss-Prot + TrEMBL proteins) database by Kriventseva et al. [10] offers an automatic classification of Swiss-Prot and TrEMBL proteins into groups of related proteins. The clustering approach is based on two steps. First, a similarity matrix of 'all-against-all' comparisons of the protein sequences is built. The similarity matrix is computed using the Smith-Waterman algorithm. A Monte-Carlo simulation, resulting in a Z-score is used to estimate the statistical significance of similarity between potentially related proteins. Then clusters are built using single link algorithm for different levels of protein similarity. Only clusters which contain more than one protein are presented in the database.

CLICK(CLuster Identification via Connectivity Kernels) by Sharan and Shamir [11] was originally developed for grouping of genes with similar expression patterns into clusters, but it is applicable as well to other biological problems. The algorithm uses graph theoretic and statistical techniques to identify tight groups of highly similar elements(kernels), which are likely to belong to the same true cluster. Several heuristic procedures are then used to expand the kernels into full clustering.

Krause [12] has used set-theoritic and single link clustering approaches for constructing the phylogeny tree of protein sequences.

Hong et al. [13] have designed a hypergraph based model for clustering data in a high dimensional space. A set of association rules are formed to find the frequent item sets of Expressed Sequence Tags(ESTs) and are used to construct the hypergraph.

SCOP(structural Classification of Proteins) database by Conte et al. [14] provides a description of the relationships of known protein structures. Proteins are classified to reflect both structural and evolutionary relatedness. The hierarchical classification has three levels; the first two levels (family and superfamily) describe near and distant evolutionary relationships; the third (fold) describes geometrical relationships.

Guralnik and Karypis [15] have designed a K-means based algorithm for clustering protein sequences by using frequently occurring patterns. Bolten et al. [16] have used transitive homolgy, a graph theoritic based approach for clustering protein sequences for structure prediction. Somervuo and Kohonen [17] have used self organizing map (SOM) for clustering the protein sequences and determining the prototypes.

Single link clustering method is computationally very expensive for clustering a 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. SOM is also computationally very expensive for protein sequence clustering.

Here, we propose a method to extend the leader algorithm as it is a simple incremental clustering algorithm with a time complexity of $O(nd)$ (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 is an incremental hierarchical clustering technique suitable for both numerical and sequence data sets.

Threshold values differ from domain to domain and they are different for different data sets. Threshold values should be chosen properly for every data set by using the distance measures (Euclidean distance for numerical data sets and similarity score for protein sequences). For numerical data sets, Euclidean distance between the intracluster patterns is smaller than the intercluster patterns. Two patterns with numerical attributes belonging to the same cluster or group have a smaller Euclidean distance between them. For protein sequence data sets, a protein sequence alignment algorithm with a substitution matrix and appropriate gap penalties are used for finding the similarity score. Similarity score between intracluster patterns is larger than that between intercluster patterns. Two protein sequences belonging to the same cluster or group have a larger similarity score between them. Euclidean distance is a measure of dissimilarity index and similarity score is a measure of similarity index. Though the clustering principle is same the programs are different for these two types of data sets because of the different distance measures and their definitions.

The proposed method and the experimental results are discussed in the following sections.

3 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 threshold value. In this method, the first pattern is selected as the leader of a cluster and each of the remaining patterns are processed depending on the existing leaders (Lds) and 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/subclusters in the i^{th} cluster as shown in Figure 2. 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. This is a kind of divisive clustering approach. A two level hierarchical structure as shown in Figure 3 can be generated using only two database scans.

There would be several subgroups/subclusters in each group/cluster of protein sequences. It is necessary to find these subgroups/subclusters also. If a representative from each subgroup is chosen then naturally classification accuracy would be improved. For choosing the threshold and subthreshold values, we are using the following procedure. Threshold and subthreshold values can be chosen depending on the maximum and the minimum similarity score values between the objects of a class in case of supervised learning. Incase of a unsupervised learning technique, similarity score values are calculated from a sequence to all the other sequences in the training data and they are sorted in the descending order. Threshold values can be chosen properly by observing the largest score values that are closer to each other in the beginning of the sorted list which may correspond to the sequences belonging to the same cluster as that of the sequence with which the similarity score values were computed. Threshold value should be chosen properly depending on the number of clusters to be generated. If the threshold value is too large then a large number of clusters are generated and if the threshold value is too small then very few clusters are generated. Subthreshold value should be larger than the threshold value to group the objects of a subgroup/subcluster. Prototypes (representatives of the clusters and subclusters) are generated using the training data set. During classification/testing phase, for every test pattern of the testing data set, the nearest leader is found first and then the nearest subleader in that cluster is determined. Then the test pattern is classified based on the nearest of these two. For different threshold and subthreshold values, experiments (both training and testing) are conducted and the results are evaluated. To evaluate the clustering quality (quality of the prototypes selected) labelled patterns are considered. During training phase they are treated as unlabelled patterns and prototypes are selected. The quality of the prototypes is evaluated using the CA obtained for the testing data set. Leaders-Subleaders algorithm require two database scans ($h = 2$) and its time complexity is $O(ndh)$ and is computationally less expensive compared to most of the other partitional and hierarchical clustering algorithms as shown in Table 1. Space complexity of Leaders-Subleaders algorithm is $O((L + SL)d)$, where L and SL are the total number of leaders and subleaders respectively. For a h level hierarchical structure, the space complexity is

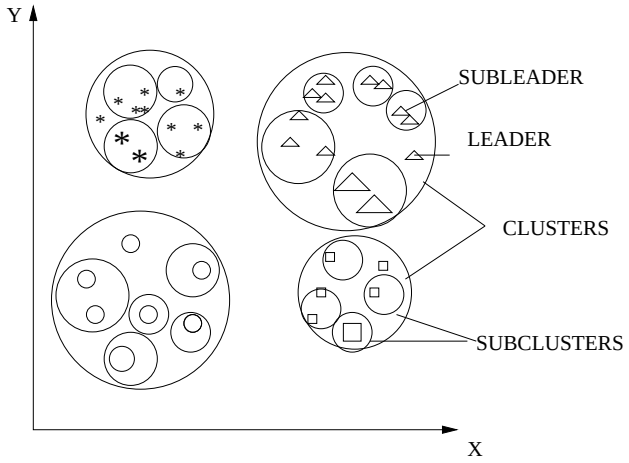

Figure 2: Clusters in Leaders-Subleaders algorithm

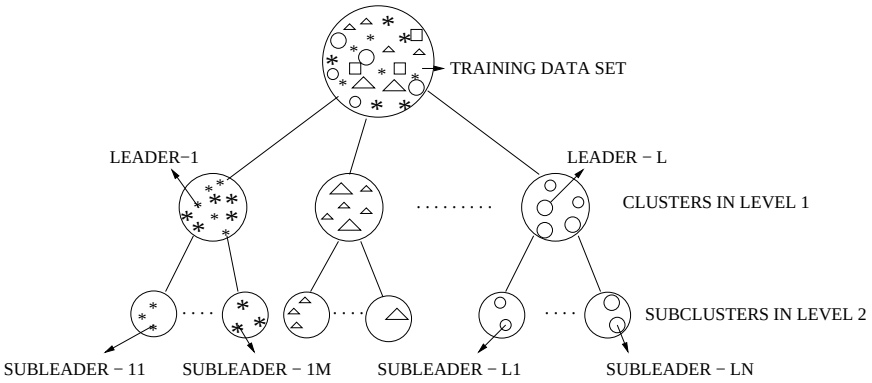

Figure 3: Hierarchical structure in Leaders-Subleaders algorithm

$O((\sum_{j=1}^{h}(L_j))d)$, and the sum of the prototypes is less than the total number of patterns - n. 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, classification time is less as only part of the hierarchical structure is searched during the testing phase. 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.

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

Leader algorithm :

Training:

1. *Select threshold value*
2. *Initialize a leader and add it to leader list and set leader counter, $L = 1$*
3. *Do 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. *Do 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, Do for i = 1 to L $\{SL = SL + SL_i\}$*

Algorithm	Scheme	Data sets	Time Complexity	Space Complexity
Single/Complete Link	Hie	Num/Seq	$O(n^3d)$	$O(n^2)$
K-means	Par	Num	$O(nKdt)$	$O(Kd)$
K-medoids(PAM)	Par	Num/Seq	$O(K(n-K)^2d)$	$O(Kd)$
DBSCAN	Hie/Par	Num/Seq	$O(n(logn)d)$	$O(nd)$
CURE	Hie	Num/Seq	$O(N^2_{sample}d)$	$O(N^2_{sample})$
BIRCH	Hie	Num	$O(nd)$	$O(NdB)$
Leader	Par	Num/Seq	$O(nd)$	$O(Ld)$
Proposed method: Leaders-Subleaders	Hie	Num/Seq	$O(nd)$, $h=2$	$O((L+SL)d)$
For h level			$O(ndh)$	$O((\sum_{j=1}^{h}(L_j))d)$

Table 1: Time and space complexities of the well known clustering approaches

(n: total # patterns, d: dimensionality of the pattern, K: # prototypes, t: # iterations, h: # levels/db scans, N: # nodes, N_{sample} = # samples, B: # Branching factor, L: # Leaders, SL: # Subleaders, Num: Numerical, Seq: Sequence, Hie: Hierarchical and Par: Partitional)

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*

Table 1 gives the time and space complexities of the well known clustering algorithms and also the proposed method. Also the type of the data sets on which these algorithms can be applied and type of the clustering scheme (hierarchical/partitional) are also mentioned. The experimental results of the proposed method are discussed in the following section.

4 Experimental Results

To evaluate the performance of Leader and Leaders-Subleaders algorithms, the following two data sets were considered. The data sets are in the FASTA format. The local alignment program provided by Huang and Miller [18] is used with necessary modifications for finding the similarity score. PAM250 scoring matrix and affine gap penalties are used in the program for calculating the similarity score.

Protein sequence data set 1:

In this data set, protein sequences from three families in which the sequences have been classified into subfamilies/subgroups depending on their functions have been considered. Protein sequences of HLA protein family have been collected from www.obi.ac.uk/imgt/hla. It contains 1609 sequences grouped into

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-Sea-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-Sea-Lds-Sublds	350	750	68	835	1303.91	2720.98	98.76
	400	800	109	816	1663.26	2776.19	98.91

Table 2: Comparison of time and classification accuracy for data set 1

(Lds: Leaders, Sublds: Subleaders, Seq-Sea: Sequential Search)

Algorithm	Thres-hold	Subthr-eshold	Lds	Sublds	Training time (secs)	Testing time (secs)	Classification Accuracy (%)
Nearest Neighbour	-	-	-	-	-	45125.12	97.93
Leader	100	-	110	-	2506.64	1521.85	96.34
	200	-	220	-	4481.40	2495.38	96.34
	300	-	473	-	8272.21	4575.88	96.34
Seq-Sea-Lds	100	-	110	-	2506.64	4395.23	97.84
	200	-	220	-	4481.40	4465.82	98.03
	300	-	473	-	8272.21	7014.60	97.93
Leaders-Subleaders	100	300	100	431	2663.09	1602.20	97.37
	200	400	220	719	4583.64	2604.21	97.46
Seq-Sea-Lds-Sublds	100	300	100	431	2663.09	2249.01	97.37
	200	400	220	719	4583.64	2836.28	97.46

Table 3: Comparison of time and classification accuracy for data set 2

(Lds: Leaders, Sublds: Subleaders, Seq-Sea: Sequential Search)

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.

Protein sequence data set 2:

In PROSITE database, profiles of different domains/groups/families based on the conserved regions or motifs have been stored. The protein sequences belonging to these domains mostly have similar functions or three dimensional structures. Protein sequences from Swiss-prot and TrEMBL database which corresponds to each of these PROSITE groups have been accessed and stored to form the database by using the sequence retreival system from http://tw.expasy.org. Thus, totally we have considered 29 different classes/groups containing the sequences corresponding to PROSITE groups. The data set considered has totally 4325 sequences. From this, randomly 3259 sequences were selected for training and 1066 for testing.

The experiments were done on an 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 tables 2 and 3), it is evident that both the algorithms performed well compared to the nearest neighbour approach [19] 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 in Leader algorithm (Seq-Sea-Lds) or subcluster in Leaders-Subleaders algorithm (Seq-Sea-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-Sea-Lds and Seq-Sea-Lds-Sublds algorithms [20]. Even then, it is less compared to NNC. Seq-Sea-Lds algorithm requires more testing time as it searches an entire cluster and its accuracy can be as that of NNC. Also Seq-Sea-Lds algorithm requires more memory during testing phase compared to Seq-Sea-Lds-Sublds algorithm.

Time complexity of the local alignment algorithm is quadratic in nature; it is $O(uv)$ where u and v are the lengths of the 2 sequences to be compared. We have used sequence alignment score for clustering or classification as it gives higher accuracy when compared to clustering/classification methods using motifs or frequent item sets [13, 15] (training time is too high but classification time is less). Because of huge computation time required for the protein sequence clustering or classification time using sequence alignment, it takes **months** together to get results for NNC or other methods if experiments are conducted for the entire protein database. Therefore we have reported the results for medium size data sets.

Though the results are given here for medium size data sets (because of the huge computation time requirements), the proposed algorithm is applicable for large data sets. The results presented are for the case where training data could be accomodated in the main memory. However, the disk I/O time increases during training/design phase if the entire training data cannot be stored in the main memory. The latest computers have a RAM capacity of about 4 GB or more. It is possible to accomodate the entire training set in such computers. The proposed algorithm can be easily implemented if RAM size is capable of holding the entire training set. Otherwise few more disk read/write operations and creation of some intermediate files in the disk are involved if it is to be run on a computer having less RAM space. Analysis of classification time for both the cases is given in the following subsection.

4.1 Analysis of Classification Time

Let n be the number of training patterns and m be the number of test patterns. Let b_n and b_m be the number of blocks to be read from the disk for reading n and m patterns respectively into the main memory. Let b_L and b_{LS} represents the number of blocks to be read in case of Leader and Leaders-Subleaders algorithms consisting of the prototypes generated in these two algorithms. Here $n > m$, $b_n > b_m$ and $b_n > b_{LS} > b_L$. Let us consider the following two cases for analysing the classification time.

Case 1: n training patterns can be accomodated in the main memory.

In the following algorithms, prototypes are stored in the main memory but test patterns need not be stored during testing phase.

NNC: NNC does have any training phase. During testing phase, b_n blocks are read from the disk for reading n patterns into the main memory. Totally b_m blocks are read from the disk for reading m test patterns. Each of the test pattern is compared with all the n training patterns to find the nearest neighbour to classify the test pattern. This involves a total random memory access time of mn units.

Leader: During training phase, b_n blocks are read from the disk and the Leaders generated are written to a file which requires say, b_L blocks. During testing phase, b_L and b_m blocks are read and the random memory access time for clssification is mL units.

Seq-Sea-Lds: During training phase, b_n blocks are read from the disk and the Leaders generated are written to a file which requires say, b_L blocks. Intermediate structure as shown in level-1 of Figure 3 is stored in the disk (which is around b_n blocks write operation). During testing phase, b_L, b_{Lb} (the blocks

Algorithm	Training/Design Phase Disk I/O	Testing Phase Disk I/O	Testing Phase RAM Access
NNC	–	$b_n + b_m$	mn
Leader	$b_n + b_L$	$b_L + b_m$	mL
Seq-Sea-Lds	$2b_n + b_L$	$b_L + b_{Lb} + b_m$	$m(L + Lb)$
Leaders-Subleaders	$b_n + b_{LS}$	$b_{LS} + b_m$	$m(L + SL_i)$
Seq-Sea-Lds-Sublds	$2b_n + b_{LS}$	$b_{LS} + b_{SLb} + b_m$	$m(L + SL_i + SLb)$

Table 4: Disk I/O and RAM timings in training and testing phases for case 1

containing the selected leader/cluster members) and b_m blocks are read and the random memory access time for clssification is $m(L + Lb)$ units, where Lb is the total members in the selected leader/cluster.

Leaders-Subleaders: During training phase, b_n blocks are read from the disk and the Leaders and subleaders generated are written to a file which requires say, b_{LS} blocks. During testing phase, b_{LS} and b_m blocks are read and the random memory access time for clssification is $m(L + SL_i)$ units (where SL_i corresponds to the subleaders/subclusters in the i^{th} cluster).

Seq-Sea-Lds-Sublds: During training phase, b_n blocks are read from the disk and the Leaders and Subleaders generated are written to a file which requires say, b_{LS} blocks. Intermediate structure as shown in level-1 and level-2 of Figure 3 is stored in the disk (which is around b_n blocks write operation). During testing phase b_{LS}, b_{SLb} (the blocks containing the selected subleader/subcluster members) and b_m blocks are read and the random memory access time for clssification is $m(L + SL_i + SLb)$ units, where SLb is the total members in the selected subleader/subcluster.

Case 2: n training patterns cannot be accomodated in the main memory.

In the following algorithms, prototypes are stored in the main memory but test patterns need not be stored during testing phase except for NNC.

NNC (efficient): NNC does not have any training phase. Let us assume that m test patterns can be accomodated in the main memory. During testing phase, b_m blocks are read to store m patterns in the main memory. Then b_n blocks are read one at a time and all the m test patterns are compared with the training patterns in the block presently read and the nearest neighbour information is updated for all the m test patterns. The test patterns are classified after b_n blocks are read. This efficient NNC involves a total random memory access time of mn units.

Leader: During training phase, b_n blocks are read from the disk and the Leaders generated are written to a file which requires say, b_L blocks. During testing phase, b_L and b_m blocks are read and the random memory access time for clssification is mL units.

Seq-Sea-Lds: During training phase, b_n blocks are read from the disk and the Leaders generated are written to a file which requires say, b_L blocks. Intermediate structure as shown in level-1 of Figure 3 is stored in the disk (which is around b_n blocks write operation). Let p be the number of times the b_n blocks are accessed during training phase. During testing phase, b_L, b_{Lb} (the blocks containing the selected leader/cluster members) and b_m blocks are read and the random memory access time for clssification is $m(L + Lb)$ units, where Lb is the total members in the selected leader/cluster.

Leaders-Subleaders: During training phase, b_n blocks are read from the disk and the information about the intermediate structure consisting of Leaders/clusters and their members at level-1 (as shown in Figure 3) are written to a file which requires say, b_n blocks. These are read again to main memory for forming subleaders at level-2 and the information about this intermediate structure consisting of level-1 and level-2 in Figure 3 are written to a file which corresponds to say, b_n blocks, which may be used for processing in the next level. Let q be the number of times the b_n blocks are accessed during training phase. Information about Leaders and Subleaders (b_{LS} blocks) are written to disk for classification purpose. During testing phase, b_{LS} and b_m blocks are read and the random memory access time for clssification is $m(L + SL_i)$ units (where SL_i corresponds to the subleaders/subclusters in the i^{th} cluster).

Seq-Sea-Lds-Sublds: During training phase, b_n blocks are read from the disk and the information

Algorithm	Training/Design Phase Disk I/O	Testing Phase Disk I/O	Testing Phase RAM Access
NNC (efficient)	–	$b_n + b_m$	mn
Leader	$b_n + b_L$	$b_L + b_m$	mL
Seq-Sea-Lds	$pb_n + b_L$	$b_L + b_{Lb} + b_m$	$m(L + Lb)$
Leaders-Subleaders	$qb_n + b_{LS}$	$b_{LS} + b_m$	$m(L + SL_i)$
Seq-Sea-Lds-Sublds	$qb_n + b_{LS}$	$b_{LS} + b_{SLb} + b_m$	$m(L + SL_i + SLb)$

Table 5: Disk I/O and RAM timings in training and testing phases for case 2

about the intermediate structure consisting of Leaders/clusters and their members at level-1 (as shown in Figure 3) are written to a file which requires say, b_n blocks. These are read again to main memory for forming subleaders at level-2 and the information about this intermediate structure consisting of level-1 and level-2 in Figure 3 are written to a file which corresponds to say, b_n blocks, which may be used for processing in the next level. Let q be the number of times the b_n blocks are accessed during training phase. Information about Leaders and Subleaders (b_{LS} blocks) are written to a file for classification purpose. Leaders and Subleaders generated are written to a file which corresponds to say, b_{LS} blocks. Intermediate structure as shown in level-1 and level-2 of Figure 3 is stored in the disk (which is around b_n blocks write operation). During testing phase b_{LS}, b_{SLb} (the blocks containing the selected subleader/subcluster members) and b_m blocks are read and the random memory access time for clssification is $m(L + SL_i + SLb)$ units, where SLb is the total members in the selected subleader/subcluster.

From tables 4 and 5 and the analysis given above, the following conclusion is clear.

If all the n training patterns cannot be accomodated in the main memory, then disk I/O time increases in the training/design phase in Seq-Sea-Lds, Leaders-Subleaders and Seq-Sea-Lds-Sublds algorithms. Training or design phase is only once and what is important is the testing time. As prototypes can be accomodated in the main memory, disk I/O time would not increase during testing phase for Leader, Seq-Sea-Lds, Leaders-Subleaders and Seq-Sea-Lds-Sublds algorithms. Disk I/O time would not increase in the testing phase in NNC if it is designed efficiently (if all the test patterns can be accomodated in the main memory).

5 Conclusions

In this paper, experimental results of Leaders-Subleaders algorithm on protein sequence data sets 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 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 family and subfamily relationships in protein sequences. We further like to compare the classification accuracy and computation time with few more algorithms. The proposed technique can also be used on numerical data sets [21, 22], text and web document collection with appropriate distance measures and hence with necessary modifications to the algorithms.

References

[1] P. Clote and R. Backofen, *Computational Molecular Biology - An Introduction.* John Wiley & Sons, Ltd., 2000.

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

[3] T. F. Smith and M. S. Waterman. *Identification of Common Molecular Subsequences.* J. of Mol. Biology, Vol 147, pp. 195-197, 1981.

[4] 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, Vol 48, pp. 443-453, 1970.

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

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

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

[8] H. Spath, *Cluster Analysis Algorithms for Data Reduction and Classification.* Ellis Horwood, Chichester, UK, 1980.

[9] G. Yona, N. Linial and M. Linial. *ProtoMap: Automatic Classification of Protein Sequences and Hierarchy of Protein Families.* Nucleic Acids Research, Vol 28, 2000.

[10] E. V. Kriventseva, W. Fleischmann, E. M. Zdobnov and G. Apweiler. *CluSTr: a Database of Clusters of SWISS-PROT+TrEMBL Proteins.* Nucleic Acids Research, Vol 29, 2001.

[11] R. Sharan and R. Shamir. *CLICK: A clustering Algorithm with Applications to Gene Expression Analysis.* In Proc. of 8^{th} ISMB, 2000.

[12] A. Krause. *Large Scale Clustering of Protein Sequences.* Ph.D. Dissertation, Berlin, 2002.

[13] E. 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, Vol 28, 2000.

[15] V. Guralnik and G. Karypis. *A Scalable Algorithm for Clustering Sequential Data* In Proc. of I^{st} IEEE conference on Data Mining, 2001.

[16] E. Bolten, A. Schliep and S. Schneckener. *Clustering Protein Sequences-Structure Prediction by Transitive Homology.* GCB, 1999.

[17] P. Somervuo, T. Kohonen. *Clustering and Visualization of Large Protein Sequence Databases by Means of an Extension of the Self-Organizing Map.* In Proc. of 3^{rd} Int. Conf., Discovery Science, pp. 76-85, 2000.

[18] X. Huang and W. Miller. *A Time-Efficient, Linear-Space Local Similarity Algorithm.* Advances in Applied Mathematics, Vol 12, pp. 337-357, 1991.

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

[20] P. A. Vijaya, M. N. Murty and D. K. Subramanian. *An Efficient Incremental Protein Sequence Clustering Algorithm.* In Proc. of IEEE Int. Conf. on Convergent Technologies, Asia Pacific, Vol 1, Bangalore, India, pp. 409-413, 2003.

[21] P. A. Vijaya, M. N. Murty and D. K. Subramanian. *An Efficient Incremental Clustering Algorithm for Large Data Sets.* In Proc. of 2^{nd} ICAAI-2003, Kolhapur, India, pp. 79-85, 2003.

[22] P. A. Vijaya, M. N. Murty and D. K. Subramanian. *Leaders-Subleaders: An Efficient Hierarchical Clustering Algorithm for Large Data Sets.* Pattern Recognition Letters, Vol 25, pp. 503-511, 2004.

P. A. Vijaya received her B.E. degree in E & C Engg. from Malnad College of Engg., Hassan, University of Mysore, India and M.E. degree in Computer Science and Engg. from the Dept of Computer Science and Automation, Indian Institute of Science, Bangalore, India. She is a faculty in the Dept. of E & C Engg., MCE, Hassan. Currently, she is doing her Ph.D programme in the Dept. of CSA, I.I.Sc., Bangalore. Her research area is pattern clustering and classification.

M. Narasimha Murty received his B.E., M.E. and Ph.D. degrees from Indian Institute of Science, Bangalore. Currently, he is a Professor in the Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India. His research interests are in pattern recognition, data mining, genetic algorithms and machine learning techniques.

D. K. Subramanian received his B.E., M.E. and Ph.D. degrees from Indian Institute of Science, Bangalore. Currently, he is a Professor in the Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India. His research interests are in distributed databases, transaction processing and data mining.