Pattern Anal Applic (2006) 9:243–255
DOI 10.1007/s10044-006-0040-z

THEORETICAL ADVANCES

Efficient median based clustering and classification techniques for protein sequences

P. A. Vijaya · M. Narasimha Murty · D. K. Subramanian

Received: 30 October 2004 / Accepted: 6 July 2006 / Published online: 22 August 2006
© Springer-Verlag London Limited 2006

Abstract In this paper, an efficient K-medians clustering (unsupervised) algorithm for prototype selection and Supervised K-medians (SKM) classification technique for protein sequences are presented. For sequence data sets, a median string/sequence can be used as the cluster/group representative. In K-medians clustering technique, a desired number of clusters, *K*, each represented by a median string/sequence, is generated and these median sequences are used as prototypes for classifying the new/test sequence whereas in SKM classification technique, median sequence in each group/class of labelled protein sequences is determined and the set of median sequences is used as prototypes for classification purpose. It is found that the K-medians clustering technique outperforms the leader based technique and also SKM classification technique performs better than that of motifs based approach for the data sets used. We further use a simple technique to reduce time and space requirements during protein sequence clustering and classification. During training and testing phase, the similarity score value between a pair of sequences is determined by selecting a portion of the sequence instead of the entire sequence. It is like selecting a subset of features for sequence data sets. The experimental results of the proposed method on K-medians, SKM and Nearest Neighbour Classifier (NNC) techniques show that the Classification Accuracy (CA) using the prototypes generated/used does not degrade much but the training and testing time are reduced significantly. Thus the experimental results indicate that the similarity score does not need to be calculated by considering the entire length of the sequence for achieving a good CA. Even space requirement is reduced during both training and classification.

Keywords Clustering · Protein sequences · Median strings/sequences · Set median · Prototypes · Feature selection · Classification accuracy

P. A. Vijaya (✉) · M. N. Murty · D. K. Subramanian
Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560012, India
e-mail: pav@csa.iisc.ernet.in

M. N. Murty
e-mail: mnm@csa.iisc.ernet.in

D. K. Subramanian
e-mail: dks@csa.iisc.ernet.in

1 Originality and contribution

Median string for a set of strings has been defined in [17] and [20]. Self Organizing Map (SOM) method has been used by Somervuo and Kohonen [39] for clustering protein sequences and to determine the generalized median strings using similarity values between the protein sequences (using FASTA method with BLOSUM50 scoring matrix [29]). SOM is computationally very expensive for protein sequences. Martinez et al. have determined the performance of k-Nearest Neighbour Classifier (k-NNC) using the prototypes derived using approximated median string and set median [20]. They have experimented on chromosome data using edit distance (a dissimilarity index measure). We use median sequence/string for a set of sequences (set median) based on the pairwise similarity score which is a measure of similarity index. In this paper, supervised and unsupervised K-medians clustering algorithms for a set of protein sequences based on the

similarity scores obtained using a pairwise sequence alignment (local alignment) algorithm are proposed for prototype selection for protein sequence classification. Performance of the K-medians clustering (unsupervised) algorithm [Classification Accuracy (CA) obtained using the prototypes selected] is compared with that of leader based technique and Supervised K-medians (SKM) classification technique is compared with that of motifs based technique and Nearest Neighbour Classifier (NNC) [5]. K-medians clustering algorithm outperforms the leader based technique and SKM classification technique performs better than that of motifs based approach in terms of CA and is computationally less expensive when compared to NNC. Performance of hierarchical agglomerative clustering schemes such as Single Link Algorithm (SLA) and Complete Link Algorithm (CLA) and classification techniques such as NNC, k-NNC and Efficient-NNC on the data sets used are also reported in this paper for comparison purpose. We also use a kind of feature selection technique to reduce the time and space requirements during protein sequence clustering and classification. During training and testing phase, the similarity score value between a pair of sequences is determined by selecting a portion of the sequence instead of the entire sequence. It is like selecting a subset of features for sequence data sets and is one of the simplest feature selection scheme. The experimental results based on the proposed method of feature selection show that the CA using the prototypes generated/used does not degrade much but the training and testing time are reduced significantly. Also the CA is not affected much when the lengths of the sequences to be compared are reduced in median based algorithms. Both the time and space requirements are reduced during the run time of training and testing phase. Even testing time in NNC can be significantly reduced by using this feature selection approach without much degradation in the CA.

2 Introduction

In bioinformatics, the number of protein sequences is now more than a million. Sequence alignment is useful for discovering functional, structural and evolutionary information in biological sequences. Protein sequences that are very much alike or similar may probably have a similar biochemical function or three dimensional structure [6, 25, 30]. New sequences can be classified using sequence similarity to a known protein class/family/group. This in turn may help in predicting the protein function or secondary structure of the unknown sequence so that the expense on the biological experiments can be saved. Additionally, if two sequences from different organisms are similar, there may have been a common ancestor sequence, and the sequences are then defined as being homologous. The problem we have considered here is: Given a set of protein sequences, find a good set of prototypes to correctly classify the new/test sequence in a reasonable time.

Conventional NNC [4, 5] is computationally very expensive to classify a new/test sequence to a protein group/class. Clustering is basically an unsupervised learning technique to divide a collection of patterns into groups of similar objects using distance/similarity measures. The objective is to form *clusters* or ''natural groupings'' [5, 28] of the input patterns and hence the technique is called as *pattern clustering*. The members of the clusters and their representatives are assigned class labels. Cluster representatives or some members of the clusters can be used to assign a class label to a new pattern. Thus *pattern clustering* is used as a preprocessing step in *pattern classification* [5]. In pattern clustering or pattern classification terminology, *prototype* is a representative of a group of patterns which are closer to each other. The reduced set of a given data set also forms a set of prototypes for pattern classification. Prototype selection is primarily effective in improving the classification performance of NNC and also partially in reducing its storage and computational requirements. *Prototype selection* for pattern classification refers to the process of finding the representative elements (patterns) from the given training data set. Prototypes may be either the elements/patterns of the data set or new elements/patterns formed by analyzing the patterns in the given data set. We can use both supervised and unsupervised learning techniques for prototype selection. In case of labelled patterns, normally the given data set is separated into training set and validation/test set. Either supervised learning or unsupervised learning technique is used for finding the prototypes from the training set. The patterns of the validation set are classified based on these prototypes and the CA (CA is the ratio of the number of test patterns correctly classified to the total number of test patterns and is expressed in percentage) is determined to evaluate the quality of the prototypes selected. Cluster representatives/prototypes may be either the elements/patterns of the data set or new elements/patterns formed using the patterns in the data set. The prototypes selected are used for classifying new patterns later. We use clustering techniques (both supervised and unsupervised) to generate hard partitions [23, 28] as we are interested in prototype selection for

pattern classification. We use a set of labelled patterns (well classified sequences by experts) to evaluate the performance of the algorithms proposed in this paper. In supervised classification technique, we use the knowledge of labels while determining the prototypes whereas in case of K-medians clustering technique, we do not use labels while determining the prototypes from the training data set.

Some of the well known clustering approaches [13, 14, 23, 28] have been used for grouping protein sequences for various purpose. ProtoMap [47] is designed based on weighted directed graph. A graph theoretic approach, transitive homology, is used in [2]. In CLICK [36], graph theoretic and statistical techniques have been used. In SCOP [3] and CluSTr [19], single link clustering method [13] has been used. Self Organizing Map (SOM), based on generalized median strings is used in [39] on a set of protein sequences using a measure of similarity index.

PFAM [48] is a large collection of multiple sequence alignments and hidden Markov models covering many common protein domains. SYSTERS [49, 18] is a protein sequence database created using set theoretic concepts and single link clustering method. Protonet [50] provides global classification of proteins into hierarchical clusters. PIRSF [51], the PIR Superfamily/Family concept, the original classification based on sequence similarity, has been used as a guiding principle to provide non-overlapping clusters of protein sequences to reflect their evolutionary relationships.

In [41, 42], an efficient, incremental, top-down clustering algorithm has been used to generate a hierarchical structure of prototypes for protein sequence classification. Single link clustering method is computationally very expensive for a large set of protein sequences as it requires an all-against-all initial analysis. Also the number of database (db) scans increases if the proximity matrix [13] cannot be accommodated in the main memory. Even in graph based approaches, the distance matrix values are to be calculated. SOM is computationally more expensive when compared to other partitional clustering schemes based on K-means and leader algorithms [14, 13]. Martinez et al. [20] have determined the performance of k-NNC using the prototypes derived using approximated median string and set median. They have experimented on chromosome data using the edit distance measure [6, 25, 30] which is a measure of dissimilarity index. Han et al. [10] have designed a hypergraph based model using frequent item sets for clustering data. Guralnik and Karypis [8] have designed a K-means based algorithm using frequent itemsets. Bandyopadhyay [1] has used a feature extraction and fuzzy clustering technique for classification of amino acid sequences. Here we propose a Supervised K-medians classification technique and a one-level partitional clustering method based on set median—'K-medians', for prototype selection for protein sequence classification, as centroid cannot be defined for a set of protein sequences without determining a set of motifs.

NNC [4] is used in [46, 34] for protein secondary structure prediction. To classify a new sequence, conventional NNC method [4] is computationally very expensive for a large training data set. Wang et al. [45] have used a motifs based method and a blocks based approach is used in [11] for protein sequence classification. Supervised K-medians (SKM) algorithm [43] performs well for protein sequence classification when compared to the motifs based approach [45].

In all the clustering and classification methods involving sequence similarity, pairwise sequence similarity is calculated by considering the entire sequence and is computationally expensive. In this paper, we propose a method to reduce both the run time and space requirements without much degradation in the CA, by using a type of feature selection method. This can be used in any clustering or classification algorithm based on similarity scores of pairwise sequence alignment. We have used this in supervised and unsupervised K-medians clustering algorithms and also compared with the NNC.

This paper is organized as follows. Section 2 deals with the significance of protein sequence alignment. Section 3 contains the details of the algorithms used. Experimental results are discussed in Sect. 4. Conclusions are provided in Sect. 5.

3 Protein sequence alignment

Proteins are sequences composed of an alphabet of 20 amino acids. In protein sequences, the amino acids are abbreviated using single letter codes such as A for Alanine, S for Serine and so on [6, 25, 30]. Each amino acid is made up of three bases or nucleotides. Amino acids in a protein sequence may change after few generations because of insertion/deletion/substitution operation. Homologous sequences which are similar may have a common ancestor sequence. Similar protein sequences may have a similar biochemical function and a three dimensional structure. Similarity score between a pair of sequences is determined by aligning them using gap (-) characters. The two types of pairwise sequence alignments are local [37] and global [26]. In local alignment, stretches of sequence with the highest density of matches are aligned, thus generating

one or more islands of matches. It is suitable for sequences that differ in length or share a conserved region or domain. Local alignment helps in finding conserved amino acid patterns (motifs) in protein sequences. Local alignment programs are based on Smith–Waterman algorithm [37].

Score of an alignment A protein sequence is made up of an alphabet of 20 amino acids which are abbreviated using 20 English characters. The three edit operations are substitution (match or mismatch), insertion and deletion.

Let

$$\sum = Alphabet, \quad \sum = \{A, C, G, T, S, \ldots\}.$$

Edit operation is a pair (x,y), where

$$(x,y)\epsilon\left(\sum \bigcup \{-\}\right) \times \left(\sum \bigcup \{-\}\right), (x,y) \neq (-,-).$$

Let sequence/subsequence $a = ACCGGSA$ and sequence/subsequence $b = AGGCSG$.

One possible alignment is

$$a^0 = ACCGG - SA$$
$$b^0 = A - -GGCSG$$

The sequence of edit operations, S_e, in the above alignment is $S_e = (A,A)(C,-)(C,-)(G,G)(G,G)(-,C)(S,S)(A,G)$ (i.e. Substitution (match), Deletion, Deletion, Substitution (match), Substitution (match), Insertion, Substitution (match) and Substitution (mismatch)).

An alignment consists of pairs such as (x,y) where $x, y\epsilon(\sum \bigcup \{-\})$ such that $|a^0| = |b^0| = l_a$, where l_a is the length of the aligned sequence. So a^0 or b^0 has totally l_a characters and let a_h^0 or b_h^0 represent the hth character in a^0 or b^0.

Similarity Score (a similarity index measure) of an alignment, W, is given by

$$W(a^0, b^0) = \sum_{h=1}^{l_a} E(a_h^0, b_h^0), \tag{1}$$

where, E is the score of an operation. The 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 case of affine gap penalty, the function used is $g(f) = gap\_open\_penalty + k' \times gap\_extension\_penalty$, where k' is the contiguous number of gaps after a gap initiation/open. Dynamic programming techniques [30, 25] are used for finding the score from an optimal alignment. For finding the similarity score between two protein sequences, percent accepted mutation (PAM) or blOcks substitution matrix (BLOSUM) substitution matrices such as PAM250, BLOSUM60, BLOSUM62, etc. [25, 30] are used. PAM matrix was originally generated by Dayhoff [30, 25] using a statistical approach. PAM matrix values were derived from closely related proteins. BLOSUM was originally generated by Henikoff and Henikoff [25, 30]. They use the BLOCKS database to search for differences among sequences. PAM250 and BLOSUM62 are the most widely used substitution matrices. We use PAM250 matrix in our experiments. PAM or BLOSUM matrices contain the substitution values for all pairs of 20 amino acids. The insertion, deletion and gap penalties are to be properly selected for determining the similarity score and are suggested by the biologists. The score values for different substitutions are stored in a substitution matrix, which contains for every pair (A,B) of amino acids, an entry $\mathscr{S}_{AB}$ (the score for aligning A with B). Scores are calculated for the subsequences aligned in local alignment and for the entire aligned length in case of global alignment. Optimal alignment is the one which gives the highest similarity score value among all possible alignments between two sequences. Higher the similarity score value between a pair of sequences, they are more likely to be in the same group. Pairwise sequence alignment score value is used in clustering similar sequences or classifying a new/test sequence to a known protein class/group/family. It may further help in predicting the protein function or structure of an unknown sequence.

Edit distance measure (a dissimilarity index) can also be used to cluster similar sequences. Edit distance between a pair of aligned sequences is given by

$$\text{Edit\_dist}(a^0, b^0) = \sum_{h=1}^{l_a} e(a_h^0, b_h^0), \tag{2}$$

where, e is the distance/cost value associated with each operation. Proper distance/cost values for the various edit operations and also gap penalties are to be chosen for determining the edit distance. These values are also suggested by the biologists. Lower the edit distance value between a pair of sequences, they are more likely to be in the same group.

If u and v are the lengths of the two sequences a and b, respectively, then the time and space complexities of the local or the global alignment algorithm (dynamic programming technique) is $O(uv)$ [30, 25]. Space complexity is $O(\max(u,v))$, if the edit path is not

necessary for an a-posteriori use. Median based clustering and supervised classification techniques for protein sequence classification are explained in the next section. We use similarity score (a similarity index measure) [12] for protein sequence comparisons in our experiments.

4 Median based algorithms and timing analysis

Median string (generalized) of a set of strings can be defined as the string that minimizes the sum of distances or sum of squared distances (for a dissimilarity index measure such as Euclidean distance [14, 5] or edit distance—refer Eq. 2) or maximizes the sum of similarity scores (for a similarity index measure such as similarity score—refer Eq. 1—or Jaccard's coefficient [31]) to the strings of a given set [20, 17, 7]. Euclidean distance or Jaccard's coefficient can be determined only for input patterns with a fixed number of attributes or features, whereas edit distance or similarity scores are used for input strings of variable sizes. The generalized median string may not be a pattern present in the set of strings and the search for such a median string is a NP-Hard problem and therefore, no efficient algorithms to compute the generalized median strings can be designed [20]. Thus the use of the *set median string*, which is a string/pattern present in the set that minimizes the sum of distances (or maximizes the sum of similarity scores) to the other strings of the set is very common. Martinez et al. [20] use a greedy approach to find the *approximated median strings* which is computationally expensive when compared to that of determining the *set median strings*. We use *set median sequences/strings* in our experiments which are determined using similarity scores. To determine set median strings, K-medians algorithm which is similar to K-means algorithm is used. In both the algorithms, K patterns that are randomly selected are considered as the initial cluster representatives and are iteratively improved to minimize the sum of error/distances or squared sum of error/distances (in case of dissimilarity index measure). In K-means algorithm, the mean or the centroid in a cluster is determined at every iteration whereas in K-medians algorithm, the most centrally located pattern/string is determined. One can also use medoid as the cluster/group representative. Medoid or Median (set median) is the most centrally located pattern in a cluster but K-medoids [15, 31] and K-medians algorithms [7, 43, 44] are designed differently. Both are suitable for any type of data. A set of medoids can be determined using algorithms such as Partitioning Around Medoids (PAM) [15, 31] or Clustering LARge Applications (CLARA) [15, 31] or Clustering Large Applications with RANdomized Search (CLARANS) [27].

In PAM algorithm [15, 31], initially K patterns are arbitrarily selected as medoids and the initial partition is formed. Then every non-selected pattern is assumed to be a medoid in place of an existing medoid and the total cost incurred in swapping the objects is determined. Then that non-selected pattern is considered to be a medoid if it can reduce the total cost and minimize the intra-cluster distances. The process is carried out for t iterations or till there is no change in the clusters formed. Then finally the set of medoids which can form a good set of clusters is selected for forming the final partitions. PAM algorithm is more robust when compared to K-means algorithm [13] as it uses medoids as cluster centers and also it minimizes the sum of dissimilarities instead of sum of squared distances. PAM is suitable for any type of data set. Its time complexity is $O(K(n-K)^2\mathrm{d}t)$ and the space complexity is $O(nd)$ [27, 15], where n is the total number of patterns, d is the dimensionality of the pattern and t is the number of iterations. Time complexity of PAM algorithm is higher than that of K-medians algorithm. CLARA [15, 31] is a sampling based clustering technique designed for large data sets and it uses PAM algorithm on the randomly selected S patterns, where $S < n$, instead on the entire training set. CLARANS [27] is an improvement over PAM and CLARA and is designed for spatial databases.

We use a clustering (unsupervised) technique—'K-medians' and a classification technique—'Supervised K-medians (SKM)' for protein sequences based on the median strings/sequences. For string/sequence data sets, centroid (mean vector of a set of patterns where each pattern and centroid has d numerical attribute values) of a group/class/cluster cannot be defined without determining a set of motifs. But we can use *set median* which can be determined using sum of distances or similarity scores [20] for a set of sequences to select the group representative called the median string/sequence [39, 20, 43, 44]. Mathematically, median string/sequence of a set, s, consisting of q protein sequences (set median) using similarity score can be defined as,

$$\mathrm{Med\_Seq_s} = \mathrm{argmax}_i(\mathrm{score}_i), \tag{3}$$

where

$$\mathrm{score}_i = \sum_{j=1}^{q} W(X_i^0, Y_j^0), \quad 1 \le i \le q, \;\; i \ne j \tag{4}$$

and W is the similarity score value between the optimally aligned sequences X_i^0 and Y_j^0, where the sequence $X_i \in s$ and the sequence $Y_j \in s$.

4.1 K-medians clustering technique

In K-medians clustering algorithm, initially K sequences are randomly selected from the training set as cluster representatives and the remaining sequences are assigned to the nearest representative. The local alignment score value is calculated from a sequence to all other sequences in that cluster and the sum of these score values is determined. The sequence for which this sum is maximum is the median string/sequence of that cluster. The sequences in the training set are again assigned to the respective clusters based on the new set of median strings/sequences. This process is carried out for a fixed number of iterations or stopped when there is no change in the set of median strings. The median string/sequence of a set of protein sequences can be considered as the most centrally located pattern in that cluster and is the representative/prototype of that cluster. The median strings/sequences determined are used as prototypes for classification purpose. We call this approach as 'K-medians' algorithm as K clusters are represented by K median strings/sequences.

4.2 SKM classification technique

SKM is a classification technique [43]. In SKM algorithm, median sequence is determined in each of the protein group/class consisting of labelled patterns. Hence the value of K (total number of protein classes/groups) and the cluster elements are fixed. In SKM algorithm, median string/sequence of a class is the most centrally located pattern in that class and is the representative/prototype of that class. This is similar to Minimum Distance Classifier (MDC) [5] in which the class representative is the centroid for a set of pattern vectors, where each pattern has d numerical attributes. Testing functions are similar for K-medians and SKM algorithms but the training functions differ.

We compare the results of SKM with that of motifs based approach, NNC, k-NNC [5] and Efficient-NNC. A brief description of the motifs based classification technique has also been given in the next section along with the timing analysis. The timing analysis and the results of K-medians algorithm, leader algorithm, single link algorithm (SLA), complete link algorithm (CLA), NNC and its variants have been reported.

4.3 Feature selection method and timing analysis

4.3.1 Feature selection method for protein sequences

Selecting an appropriate portion of the sequence may be viewed as one of the simplest feature selection scheme for sequence data sets. That means the features/motifs/conserved regions present in a part of the sequence are sufficient for clustering and classification purpose. We evaluate the performance of the proposed feature selection method on NNC, K-medians and SKM algorithms. In all our algorithms, the local alignment program provided by [12] is used with necessary modifications to construct a function for finding the highest score value between two sequences from the optimal alignment. Either the entire length or the selected portion of a pair of sequences is submitted to the local alignment algorithm. We try to save time and space during the execution of this similarity score function as it is the most time and space consuming function (time and space complexity is quadratic).

4.3.2 Timing analysis

Let u and v be the lengths of the two sequences to be compared in general. Let us consider the following three cases while analysing the time requirements.

Case 1: 100% of the total length of the sequence. Let u_1 and v_1 represent these lengths for the two sequences to be compared so that $u_1 = u$ and $v_1 = v$.

Case 2: 75% of the total length of the sequence. Let u_2 and v_2 represent these lengths for the two sequences to be compared so that $u_2 = 0.75u$ and $v_2 = 0.75v$.

Case 3: 50% of the total length of the sequence. Let u_3 and v_3 represent these lengths for the two sequences to be compared so that $u_3 = 0.5u$ and $v_3 = 0.5v$.

(i) NNC, k-NNC and efficient-NNC

In NNC, there is no training/design phase. Each of the test/new patterns is compared with all the training patterns. Time and space complexity of pairwise local alignment algorithm is $O(uv)$. Therefore, the time complexity to classify a new sequence in conventional NNC is $O((uv)n)$, where n is the total number of sequences in the training set. Then the time complexity is

$O((u_1v_1)n)$, $O((u_2v_2)n)$ and $O((u_3v_3)n)$ and space complexity is $O(u_1v_1)$, $O(u_2v_2)$ and $O(u_3v_3)$ (or $O(\max(u_1,v_1))$, $O(\max(u_2,v_2))$ and $O(\max(u_3,v_3))$, if the edit path is not necessary to be remembered for later use) for the cases 1, 2 and 3, respectively. As $u_3 < u_2 < u_1$ and $v_3 < v_2 < v_1$, both the time and space requirements are reduced during run time of local alignment algorithm for the cases 2 and 3. Space complexity of NNC is $O(nl)$, if all the n sequences are stored in the main memory, where l is the average length of a sequence. Otherwise the number of database scans may increase. In k-NNC technique, the distances/similarity scores from a test pattern to all the training patterns are sorted and the k nearest neighbours are selected. Depending on the majority voting, the test pattern is classified. Time and space complexity of k-NNC are $O((uv)n + n\log n)$ and $O(nl)$ respectively. In Efficient-NNC technique, we use the following procedure to reduce the time requirements when compared to that of NNC. The pairwise similarity score is calculated initially by considering only the 50% of the total length of the sequence. If the score is above a given threshold value then the pairwise similarity score is calculated considering the entire length. We thus eliminate some of the sequences for full comparison purpose. We use this approach as the sequence data sets do not have a fixed number of attributes. This has some similarity with the efficient or modified NNC techniques [21, 24, 32, 40] proposed for data sets with a fixed number of attributes. Most of the existing efficient NNC techniques [21, 24, 32, 40, 22] cannot be applied directly on sequence data sets consisting of patterns of variable lengths and also many of these efficient NNC techniques do not scale up with the dimensionality of the pattern.

(ii) Motifs based method

For determining a set of frequently occurring motifs and sorting them, the algorithms developed by Wang et al. [45] have been used. They select a sample of the sequences (using random sampling without replacement), q_s, from a given set of sequences, q, to determine a set of motifs. They use a *generalized suffix tree* (*GST*) which is an extension of suffix tree designed for representing a set of strings [45]. The time complexity of GST is proportional to the total length of all the sequences used for finding a set of motifs [45]. Therefore, the time complexity of this GST for a group/class of sequences is $O(q_{sl})$, where l is the average length of a sequence. Motifs generated are tested with the rest of the sequences in the given group in order to find their frequency of occurrence. The most time consuming part in their algorithm [45] is that of finding the number of occurrences of each motif. Training time depends on the input parameters such as minimum length of the motif, minimum number of occurrences, number of mutations and form/style of the motif (*wxyz..* or *wx..*yz..*). We determine motifs of the form *wxyz..* for different number of mutation values in our experiments. We appropriately fix the minimum number of occurrence as a percentage of the total number of sequences in that group. The short motifs (length of 4–10 characters) have higher frequency of occurrence. The frequency of occurrence of these motifs increases as the number of mutations is increased (as reported in [45]). The total number of motifs generated increases with increase in the number of mutations. Therefore the training time also increases as the number of mutations is increased. Sorting software developed by Wang et al. [45] sorts the motifs according to their length and also eliminates the substrings whose frequency of occurrence is same as that of their superstrings. Time complexity of this sorting algorithm is at least $O(r\log r)$ [33], where r represents the average number of motifs generated in a class and for K classes, the time complexity of the sorting algorithm is $O((r\log r)K)$. Programs were developed by us for organizing the output of the sorting program (i.e. to remove all the unwanted statements and to have only a list of motifs without the character '*' and with their frequency of occurrence, for each class) and for classifying the test sequences. To organize the output of the sorting program for each class, the time complexity is $O(Kr)$ as at least r motifs are read from each class. The classification program is a simple string matching program. To find whether a given string is a substring of a sequence of length l, the time complexity is $O(l)$ [16]. Therefore, to classify a new/test sequence using Motifs based method, the time complexity is $O(fl)$, where f is the total number of selected motifs from all the classes and l corresponds to the length of the test sequence. Discovery, sorting, organization and selection of motifs are the different parts in the training phase and many intermediate files are created. Some steps in the training phase involve manual intervention and thus the procedure is semi-automatic in motifs based method. Space complexity in the motifs based algorithm is $O(ql)$ as all the q sequences of an average length of l each, belonging to a class are stored in the main memory at any time.

In the experiments, motifs were generated by inputting the values for the parameters. For classification purpose, frequently occurring motifs of size 4–10 characters were selected. If a class contains less than 10

motifs, then all the frequently occurring motifs were considered from that class irrespective of their lengths. Highest CA was obtained for protein sequence classification for this type of motif selection. CA was determined using the motifs selected with a weight of 1 and also a weight calculated based on the frequency of occurrence. But they did not differ much.

(iii) K-medians and SKM algorithms

Time and space complexity analysis is same for K-medians (unsupervised clustering) and SKM algorithms except for the number of iterations. Time complexity to find the median string for all clusters/groups is $O(q^2(uv)Kt)$, where t is the number of iterations and q is the size of a cluster (on an average, the size of a cluster may be considered as n/K). Value of t is 1 for SKM algorithm. Only $q(q-1)/2$ score values are to be calculated for each cluster/group as $W(a_i^0, b_j^0)$ is equal to $W(b_j^0, a_i^0)$. Space complexity of K-medians and SKM algorithms are $O(nl)$ (if proximity values are calculated as and when required) and $O(q^2)$ respectively. Sum of the score values are to be calculated for all the q sequences in a cluster/group and the median is to be selected. Time complexity to classify a new/test sequence is $O((uv)K)$ for both K-medians and SKM algorithms. Herein, both the time and space requirements are reduced for cases 2 and 3, during the execution of the local alignment algorithm in training as well as in testing phase. The training phase is fully automatic in the SKM method.

(iv) Leader algorithm

Leader is an incremental clustering algorithm in which L leaders representing L clusters are generated using a suitable threshold value [38]. The first pattern is selected as the leader of a cluster. The remaining patterns are assigned to one of the existing clusters or to a new cluster depending on the chosen threshold value. In leader based algorithms, threshold value T—a user defined parameter, should be properly chosen. We use the following procedure to choose the threshold value. Threshold value can be chosen depending on the maximum and the minimum similarity values between the objects of a class in case of supervised learning. In case of unsupervised learning technique, similarity values are calculated from a pattern to all the other patterns in the training data and they are sorted in the descending order. The largest similarity values that are closer to each other in the beginning of the sorted list correspond to the patterns belonging to the same cluster as that of the pattern with which the similarity values were computed. The largest value (other than $W(u^0,u^0)$ (refer Eq. 1) will give an idea to choose the threshold value. It requires only one database scan and is suitable for large data sets. Time complexity to find the leaders is $O((uv)Ln)$ and space complexity is $O(Ll)$. Time complexity to classify a new/test sequence is $O((uv)L)$.

(v) Single link and complete link algorithms

Hierarchical agglomerative clustering (bottom-up/merging) procedures start with n singleton clusters and form the hierarchy by successively merging the clusters until a desired number of clusters is obtained. Single link algorithm (SLA) and complete link algorithm (CLA) are of this type. In SLA, the distance between two clusters C_1 and C_2 is the minimum of the distances $d(X,Y)$ (a dissimilarity index), where $X \in C_1$ and $Y \in C_2$ and the two clusters with the smallest minimum pairwise distance are merged at every level. In CLA, the distance between two clusters C_1 and C_2 is the maximum of the distances $d(X,Y)$, where $X \in C_1$ and $Y \in C_2$ and the two clusters with the smallest maximum pairwise distance are merged at every level. As we use similarity index measure, we made suitable changes in the algorithm. For SLA and CLA, proximity matrix requires $O(n^2)$ space. Time complexity of SLA and CLA are $O(n^2uv)$ and $O(n^2(\log n)uv)$, respectively [14, 35]. In hierarchical clustering techniques, a cluster representative like centroid or median or medoid is chosen from the resulting clusters at the end. We use median sequence (set median) as the cluster representative. Time complexity to classify a new/test sequence in SLA or CLA is $O(uvK)$.

5 Experimental results

To evaluate the performance of the algorithms, the following two data sets were considered. The data sets are in the FASTA format [25, 30].

5.1 Protein sequence data sets

Protein sequence data set 1 (PSDS1) Protein sequences of HLA protein family have been collected from ''http://www.obi.ac.uk/imgt/hla''. It contains 1,609 sequences grouped into 19 classes. Protein sequences

of AAA protein family have been collected from ''http://www.aaa-proteins.uni-graz.at/AAA/AAA-Sequences.text''. AAA protein family sequences have been categorized into six classes according to their functions and it consists of 227 sequences. From Globins protein family, sequences have been collected from 4 different groups and 629 sequences have been considered from the data set provided along with the software package hmmer-2.2g (''http://ftp.genetics.wustl.edu/pub/eddy/hmmer/hmmer-2.2g.tar.gz''). Thus, totally we have considered 29 different classes containing the sequences according to protein functions. Similar protein sequences may have similar 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 2,565 sequences. From this, randomly 1,919 sequences were selected for training and 646 for testing, such that the samples from each class are in both the training and test sets (but mutually exclusive).

Protein sequence data set 2 (PSDS2) 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 retrieval system from ''http://www.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 4,325 sequences. From this, randomly 3,259 sequences were selected for training and 1,066 for testing, such that the samples from each class are in both the training and test sets (but are mutually exclusive).

5.2 Results and discussions

The experiments were done on an Intel pentium-4 processor based machine having a clock frequency of 1,700 Mhz and 512 MB RAM. The experimental results using various algorithms are reported in Tables 1, 2, 3, 4, 5. In the result tables, training time is the total time taken for selecting the prototypes from the training set. Training or design is done only once and once the prototypes are selected, only the testing time is to be compared between the algorithms. Testing time is the time taken for classifying all the test patterns in the validation/test set. Table 1 shows the performance of K-medians algorithm, SLA, CLA and leader based clustering techniques without feature selection, for the two protein sequence data sets used. In leader based technique, the results correspond to different threshold values chosen. In K-medians algorithm, the results correspond to the desired value of K and the number of iterations, $t = 5$. CA obtained using the median sequences as prototypes is very high compared to that of leaders. In our experiments, the proximity values are calculated as and when required in K-medians algorithm as all the n sequences are stored in the main memory, whereas in SLA and CLA, $n \times n$ proximity matrix values are calculated and stored in the beginning. If $n \times n$ proximity matrix is calculated and stored in the main memory then the training time in K-medians algorithm may be further reduced. Performance of CLA is better than that of SLA on both the data sets. CLA performs very well on the data set PSDS2 when compared with that on PSDS1. Performances of the classification techniques—NNC, k-NNC and an efficient-NNC are also reported in Table 1. k-NNC performs better than NNC and Efficient NNC techniques on both the data sets in terms of CA whereas efficient-NNC provides the same CA as that of NNC in a lesser time.

Table 2 shows the performance of SKM and motifs based technique for protein sequence classification. In SKM, the number of prototypes is equal to the number of classes in the data set used and in motifs based approach, the results have been reported for different values of mutations. CA using median sequences of the known classes is better than that of motifs based approach. CA can be further improved in SKM by selecting some supporting prototypes—SKM-SP (SKM with Supporting Prototypes) in each class using a threshold value. Supporting prototypes are the sequences which are far from the median string in a class. Training time does not increase much in SKM-SP as $q \times q$ score matrix values are stored for a class in SKM. But the testing time increases with the increase in the number of supporting prototypes.

From Tables 1, 2, 3, 4, it is evident that the CA is good in median based algorithms but the training and testing time are quite high. Hence we have tested the proposed feature selection method on median sequence based algorithms and also on NNC for comparison. Tables 3, 4, 5 show the experimental results of the algorithms with feature selection. For all the three cases, three different regions (lower, middle, upper) are selected in a protein sequence. Both training and testing time are reduced for cases 2 and 3 when compared to case 1. In K-medians algorithm, even 50% of the total length itself gives very good CA when number of prototypes generated is more (from results of

Table 1 Time and classification accuracy

Data set	Algorithm	Threshold	# prototypes	Training time (s)	Testing time (s)	CA (%)
PSDS1	K-medians	–	29	22173.36	236.62	70.89
		–	109	34504.92	1102.22	95.82
		–	454	66483.00	3635.37	99.22
		–	554	99255.67	4410.45	99.22
	Leader	188	29	306.27	219.19	41.48
		400	109	967.39	853.13	57.12
		500	454	3599.18	2594.86	66.87
		600	554	4419.34	3353.57	71.05
	SLA	–	109	53643.15	698.65	70.43
		–	554	51047.65	3486.75	75.54
	CLA	–	109	55163.03	892.94	79.25
		–	554	53125.23	3480.12	84.36
	NNC	–	1919	–	19104.02	99.84
	k-NNC	$k = 5$	1919	–	22356.45	99.86
	Efficient-NNC	300	1919	–	14568.35	99.84
PSDS2	K-medians	–	29	25487.50	376.03	82.08
		–	110	47353.06	1411.77	97.09
		–	220	62936.77	2761.29	97.27
		–	473	98534.55	5262.19	97.84
	Leader	36	29	719.37	170.69	58.06
		100	110	2506.64	1521.85	96.34
		200	220	4481.40	2495.38	96.34
		300	473	8272.21	4575.88	96.34
	SLA	–	110	76221.32	1426.12	97.93
		–	473	61601.53	5080.19	97.93
	CLA	–	110	78874.33	1395.47	98.03
		–	473	75696.42	5014.19	98.12
	NNC	–	3259	–	45125.12	97.93
	k-NNC	$k = 5$	3259	–	50427.46	98.03
	Efficient-NNC	150	3259	–	32128.28	97.93

s seconds; *CA* classification accuracy; *SLA* single link algorithm; *CLA* complete link algorithm; # iterations, $t = 5$ for K-medians

Table 2 Time and CA in SKM algorithm, SKM-SP algorithm and Motifs based method

Data Set	Algorithm	# Mutations or threshold	# Motifs used or # prototypes	Training time (s)	Testing time (s)	CA (%)
PSDS1	Motifs based	0	498	6722.56	2.58	85.60
		1	7238	9859.06	40.41	86.53
		2	20346	16288.74	103.22	86.22
	SKM	–	29	3285.55	417.94	97.05
	SKM-SP	200	79	3295.23	751.02	97.83
		400	347	3313.06	2407.18	98.29
PSDS2	Motifs based	0	2472	5462.56	21.93	89.30
		1	31165	7183.43	261.81	90.52
		2	130328	23224.82	1086.00	96.52
	SKM	–	29	4187.66	488.11	96.99
	SKM-SP	100	240	4210.53	3354.62	97.37
		200	837	4418.41	14199.53	97.37

SKM Supervised K-medians; *SKM-SP* SKM with supporting prototypes

Table 3) whereas supervised K-medians algorithm performs well when 75% of the total length is considered (from results of Table 3). In NNC, we can reduce the testing time for cases 2 and 3 when compared to case 1, without any degradation in the CA. For the data set considered, it is evident from the results shown in Table 5 that the NNC performs very well with 75% of the total length itself and also accuracy has not degraded much when 50% of the total length is considered. It can also be observed from the results that the first half or lower three fourths of a protein sequence has better (or more) features/motifs and is responsible for higher accuracy for cases 2 and 3 for the data set used. When all the features are used the CA reduces [5, 23]. With feature selection, CA increases in most of the cases as can be observed from Tables 3, 4, 5. This is because the generalization error [9] may reduce when a selected subset of features is used.

Table 3 Time and CA in K-medians algorithm with and without feature selection

Data set	% of length of the sequence	Region selected in the sequence	# prototypes	Training time (s)	Testing time (s)	CA (%)
PSDS1	100	–	109	34504.92	1102.22	95.82
	75	Lower	109	21815.42	562.91	95.97
	75	Middle	109	21409.99	568.11	95.66
	75	Upper	109	20544.82	434.64	95.66
	50	Lower	109	10277.91	277.68	94.11
	50	Middle	109	10916.26	415.76	95.20
	50	Upper	109	9741.59	304.38	93.34
PSDS2	100	–	110	47353.06	1411.77	97.09
	75	Lower	110	29402.92	1346.22	96.71
	75	Middle	110	24194.95	1087.79	96.62
	75	Upper	110	23103.93	974.70	96.24
	50	Lower	110	12264.53	398.88	94.84
	50	Middle	110	13802.29	448.45	95.77
	50	Upper	110	11805.31	434.23	93.62

Table 4 Time and CA in SKM algorithm with and without feature selection

Data set	% of length of the sequence	Region selected in the sequence	# prototypes	Training time (s)	Testing time (s)	CA (%)
PSDS1	100	–	29	3285.55	417.94	97.05
	75	Lower	29	2162.25	221.23	98.14
	75	Middle	29	2219.62	199.61	97.05
	75	Upper	29	2208.84	197.26	94.27
	50	Lower	29	1288.72	129.73	96.74
	50	Middle	29	1312.48	116.86	78.63
	50	Upper	29	1300.29	114.33	65.32
PSDS2	100	–	29	4187.66	488.11	96.99
	75	Lower	29	3077.88	299.54	95.40
	75	Middle	29	3333.04	388.48	96.24
	75	Upper	29	3346.39	393.87	95.56
	50	Lower	29	1757.32	235.08	91.36
	50	Middle	29	1815.26	227.85	91.55
	50	Upper	29	1751.01	228.28	92.12

Table 5 Time and CA in NNC with and without feature selection

Data set	% of length of the sequence	Region selected in the sequence	# prototypes	Testing time (s)	CA (%)
PSDS1	100	–	1919	19104.02	99.84
	75	Lower	1919	12276.06	100.00
	75	Middle	1919	12434.65	99.84
	75	Upper	1919	12464.78	99.53
	50	Lower	1919	7360.59	99.69
	50	Middle	1919	7460.31	99.69
	50	Upper	1919	7432.94	99.07
PSDS2	100	–	3259	45125.12	97.93
	75	Lower	3259	27851.83	97.84
	75	Middle	3259	28797.25	97.56
	75	Upper	3259	32149.18	97.65
	50	Lower	3259	19021.33	97.18
	50	Middle	3259	19261.99	97.18
	50	Upper	3259	19323.94	97.56

6 Conclusions

In this paper, the experimental results on the two protein sequence data sets used show that the K-medians clustering algorithm performs well when compared to leader based technique and SKM classification technique performs better than motifs based approach. Also the CA is not affected much when lengths of the sequences to be compared are reduced in median based algorithms and NNC. Both time and space requirements are reduced during run time of training and testing phase. This feature selection approach may be

used in any clustering/classification technique which involves pairwise sequence alignment algorithm. The training and testing time may also be reduced by using FASTA or BLAST sequence alignment programs [30, 25], which are faster than the dynamic programming techniques used for sequence alignment. Whether a protein sequence belongs to more than one domain or not may be determined/decided by using the k nearest prototypes [5] and their similarity values.

Acknowledgments We thank the anonymous reviewers for their thoughtful comments and constructive suggestions, which helped to improve the quality of this paper.

References

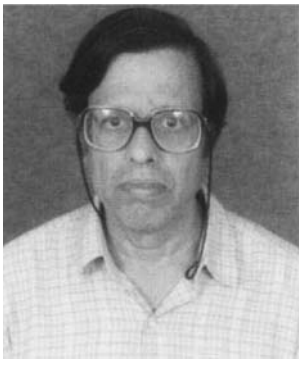
1. Bandyopadhyay S (2005) An efficient technique for superfamily classification of amino acid sequences: feature extraction, fuzzy clustering and prototype selection. Fuzzy Sets Syst 152(1):5–16
2. Bolten E, Schliep A, Schneckener S, Schomburg D, Schrader R (2001) Clustering protein sequences-structure prediction by transitive homology. Bioinformatics 17(10):935–941
3. Conte LL, Ailey B, Hubbard TJP, Brenner SE, Murzin AG, Chotia C (2000) SCOP: a structural classification of protein database. Nucleic Acids Res 28(1):257–259
4. Cover T, Hart P (1967) Nearest neighbour pattern classification. IEEE Trans Inform Theory 13(1):21–27
5. Duda RO, Hart PE, Stork DG (2000) Pattern classification, 2nd edn. Wiley, New York
6. Durbin R, Eddy S, Krogh A, Mitchison G (1998) Biological sequence analysis. Cambridge University Press, Cambridge
7. Guha S, Meyerson A, Mishra N, Motwani R, O'Callaghan L (2003) Clustering data streams: theory and practice. IEEE Trans Knowl Data Eng 153:515–528
8. Guralnik V, Karypis G (2001) A scalable algorithm for clustering sequential data. In: Proceedings of I IEEE conference on data mining, pp 179–186
9. Hamamoto Y, Uchimura S, Tomita S (1996) On the behavior of artificial neural network classifiers in high-dimensional spaces. IEEE Trans Pattern Anal Mach Intell 18(5):571–574
10. Han E, Karypis G, Kumar V, Mobasher B (1997) Clustering in a high dimensional space using hypergraph models. In: Proceedings of data mining and knowledge discovery
11. Henikoff S, Henikoff JG (1994) Protein family classification based on searching a database of blocks. Genomics 19:97–107
12. Huang X, Webb M (1991) A time-efficient, linear-space local similarity algorithm. Adv Appl Math 12:337–357
13. Jain AK, Dubes RC (1988) Algorithms for clustering data. Prentice-Hall, Upper Saddle River
14. Jain AK, Murty MN, Flynn PJ (1999) Data clustering: a review. ACM Comput Surv 31(3):264–323
15. Kaufman L, Rousseeuw P (1990) Finding groups in data: an introduction to cluster analysis. Wiley, New York
16. Knuth DE (1998) Art of computer programming, 2nd edn, vol 3. Addison- Wesley, Reading
17. Kohonen T (1985) Median strings. Pattern Recogn Lett (3):309–313
18. Krause A (2002) Large scale clustering of protein sequences. Ph.D. Thesis, Berlin
19. Kriventseva EV, Fleischmann W, Zdobnov EM, Apweiler G (2001) CluSTr: a database of clusters of SWISS-PROT+TrEMBL proteins. Nucleic Acids Res 29(1):33–36
20. Martinez CD, Juan A, Casacuberta F (2003) Median strings for k-nearest neighbour classification. Pattern Recogn Lett 24:173–181
21. MicA L, Oncina J, Vidal E (1994) A new version of the nearest-neighbor approximating and eliminating search algorithm (AESA) with linear preprocessing time and memory requirements. Pattern Recogn Lett 15:9–17
22. MicA L, Oncina J, Carrasco R (1996) A fast branch and bound nearest neighbor classifier in metric spaces. Pattern Recogn Lett 17:731–739
23. Mitra S, Acharya T (2003) Data mining: multimedia, soft computing and bioinformatics. Wiley, New York
24. Moreno F, MicA L, Oncina J (2003) A modification of the LAESA algorithm for approximated k-NN classification. Pattern Recogn Lett 22:1145–1151
25. Mount DW (2002) Bioinformatics—sequence and genome analysis. Cold Spring Harbor Lab Press, New York
26. Needleman SB, Wunsch CD (1970) A general method applicable to the search for similarities in the amino acid sequence of the proteins. J Mol Biol 48:443–453
27. Ng RT, Han J (2002) CLARANS: a method for clustering objects for spatial data mining. IEEE Trans Knowl Data Eng 14(5):1003–1016
28. Pal SK, Mitra P (2004) Pattern recognition algorithms for data mining: scalability, knowledge discovery and soft granular computing. CHAPMAN & HALL/CRC
29. Pearson W (1999) The FASTA program package http://ftp.virginia.edu/pub/fasta
30. Peter C, Rolf B (2000) Computational molecular biology—an introduction. Wiley, New York
31. Pujari AK (2000) Data mining techniques. Universities Press (India) Private Limited
32. Ramasubramanian V, Paliwal KK (2000) Fast nearest neighbor search algorithms based on approximation-elimination search. Pattern Recogn 33:1497–1510
33. Sahni S (1998) Data Structures, Algorithms and applications in C++. WCB McGraw Hill
34. Salzberg S, Cost S (1992) Predicting protein secondary structure with a nearest neighbour algorithm. J Mol Biol 227:371–374
35. Schutze H (2004) Single-link, complete-link and average-link clustering. ''http://www.csli.stanford.edu/~schuetze/completelink.html
36. Sharan R, Shamir R (2000) CLICK: a clustering algorithm with applications to gene expression analysis. In: Proceedings of 8th ISMB, pp 307–316
37. Smith TF, Waterman MS (1981) Identification of common molecular subsequences. J Mol Biol 147:195–197
38. Spath H (1980) Cluster analysis algorithms for data reduction and classification. Ellis Horwood, Chichester
39. Somervuo P, Kohonen T (2000) Clustering and visualization of large protein sequence databases by means of an extension of the self-organizing map. In: Proceedings of 3rd international conference on discovery science, pp 76–85
40. Vidal E (1986) An algorithm for finding nearest neighbors in (approximately) constant average time. Pattern Recogn Lett 4:145–157
41. Vijaya PA, Murty MN, Subramanian DK (2003) An efficient incremental protein sequence clustering algorithm. In: Proceedings of IEEE TENCON, Asia Pacific, pp 409–413
42. Vijaya PA, Murty MN, Subramanian DK (2004) An efficient hierarchical clustering algorithm for protein sequences. Int J Comput Sci Appl 1(2):61–75

43. Vijaya PA, Murty MN, Subramanian DK (2003) Supervised K-medians algorithm for protein sequence classification. In; Proceedings of 5th international conference on advanced pattern recognition, pp 129–132
44. Vijaya PA, Murty MN, Subramanian DK (2004) An efficient technique for protein sequence clustering and classification. In: Proceedings of 17th international conference on pattern recognition, Cambridge, UK, Vol II, pp 447–450
45. Wang JTL, Thomas GM, Dennis S, Bruce S (1994) Chern, discovering active motifs in sets of related protein sequences and using them for classification. Nucleic Acids Res 6(4):559–571
46. Yi TM, Eric S (1993) Protein secondary structure prediction using nearest neighbour methods. J Mol Biol 232:1117–1129
47. Yona G, Linial N, Linial M (2000) ProtoMap: automatic classification of protein sequences and hierarchy of protein families. Nucleic Acids Res 28(1):49–55
48. http://www.pfam.cgb.ki.se/
49. http://www.systers.molgen.mpg.de/
50. http://www.protonet.cs.huji.ac.il/
51. http://www.pir.georgetown.edu/pirsf/

Author Biographies

P. A. Vijaya received her B.E. degree in E and C Engineering from the Malnad College of Engineering (MCE), Hassan, University of Mysore, India and M.E. and Ph.D. degree in Computer Science and Engineering from the Department of Computer Science and Automation (CSA), Indian Institute of Science (IISc), Bangalore, India. She is an Asst. Professor in the Department of E and C Engineering, MCE, Hassan. Her research area is pattern clustering and classification.

M. Narasimha Murty received his B.E., M.E. and Ph.D. degrees from IISc, Bangalore, India. Currently, he is a Professor and also the Chairman in the Department of CSA, IISc, 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 IISc, Bangalore, India. Currently, he is Professor Emeritus in the Department of CSA, IISc, Bangalore, India. His research interests are in data mining, distributed databases and transaction processing.