An efficient technique for superfamily classification of amino acid sequences: feature extraction, fuzzy clustering and prototype selection

Sanghamitra Bandyopadhyay*

Machine Intelligence Unit, Indian Statistical Institute, 203 B.T. Road, Kolkata 700 108, India

Available online 11 November 2004

Abstract

In this article, we propose an efficient technique for classifying amino acid sequences into different superfamilies. The proposed method first extracts 20 features from a set of training sequences. The extracted features are such that they take into consideration the probabilities of occurrences of the amino acids in the different positions of the sequences. Thereafter, a genetic fuzzy clustering approach is used to automatically evolve a set of prototypes representing each class. The characteristic of this clustering method is that it does not require the a priori information about the number of clusters, and is also able to come out of locally optimal configurations. Finally, the nearest neighbor rule is used to classify an unknown sequence into a particular superfamily class, based on its proximity to the prototypes evolved using the genetic fuzzy clustering technique. This results in a significant improvement in the time required for classifying unknown sequences. Results for three superfamilies, namely globin, trypsin and ras, demonstrate the effectiveness of the proposed technique with respect to the case where all the training sequences are considered for classification using the same set of features. Comparison with the well-known technique BLAST also shows that the proposed method provides a significant improvement in terms of the time required for classification while providing comparable classification performance.

Keywords: Feature extraction; Protein classification; Position-specific information; Fuzzy clustering; Genetic algorithms; Superfamily classification

* Fax: +91 33 577 6680.

E-mail address: sanghami@isical.ac.in (S. Bandyopadhyay).

1. Introduction

Bioinformatics [5,21,27], a modern science, is basically conceptualizing biology in terms of macromolecules and applying 'informatics' techniques to understand and organize the information associated with these molecules. It primarily deals with the application of computer and statistical techniques to the management of biological information. Bioinformatics has emerged as a forefront research area in the recent past since biological data is accumulating at an accelerating rate. This is as a result of the Human Genome Project and other similar efforts, along with dramatic evolution of technology for information storage and access. In genome projects, bioinformatics includes the development of methods to search data bases quickly, to analyze DNA sequence information, and to predict protein sequence and structure from DNA sequence data. These have necessitated the development of algorithms which can extract useful information from these data bases. In response to this problem, a number of researchers have developed techniques to analyze and interpret the data, and discover concepts in the DNA, RNA and protein data bases. Classification of protein sequences into superfamilies is one of the important problems in this regard [32].

Proteins are essentially polymers of 20 different amino acids that can occur in any order. The problem of superfamily classification can be formally stated as follows [30,31]. Given an unlabeled protein (or, amino acid) sequence S and a set of known f superfamilies $F = F_1, F_2, \ldots, F_f$, we are to determine with certain degree of accuracy whether the protein S belongs to one of the superfamilies $F_i,\ i = 1, \ldots, f$ or not. Here, a superfamily is a group of proteins that share similarity in structure and function. Similar protein sequences will most probably have similar biochemical functions. Therefore, given an unknown protein, the first task is to classify it into one of the known superfamilies. This will help in predicting the protein function and/or structure of the unknown sequence; thus saving, to a large extent, the expenses incurred on expensive biological (wet) experiments in the laboratory. Perhaps, the most important practical application of such a knowledge is in drug discovery. Suppose we have obtained sequence S from some disease D and by our classification method we infer that S belongs to F_i. In order to design a drug for the disease D we may try a combination of existing drugs for F_i.

Different approaches exist for protein sequence classification [31]. Until recent days, alignment algorithms (e.g., BLAST and FASTA) [1,2,22] and Hidden Markov Models (HMM) [19] have been the major tools to help analyzing the protein sequence data and interpreting the results in a biologically meaningful manner. BLAST returns a list of *high-scoring segment pairs* between the query sequence and the sequences in the data bases. This is done after performing a sequence alignment among them. In one of the recent works [30,31] Wang et al., have tried to capture the global and local similarities of protein sequences in extracting features to be used as inputs to a Bayesian Neural network (BNN) classifier. A 2-gram encoding scheme, which extracts and counts the occurrences of two consecutive amino acids in a protein sequence, is proposed. They have also compared their technique with BLAST, SAM and other iterative methods. Although the 2-gram encoding scheme is shown to perform reasonably well, its major limitation is that it does not consider the positional significance of the residue pairs, an important consideration in superfamily classification. The number of features extracted by this scheme is also relatively large ($\geqslant 62$). This poses a serious limitation for many classification schemes. Moreover, in some recent works [30,31] the authors deal with essentially a two class classification problem, where the query sequence is classified as either belonging to a superfamily, or not. However, the importance of extracting numerical features from protein sequences has its own advantage in that it thereafter allows for the application of several low-complexity analysis algorithms like classification and clustering. Although clustering in the sequence space has been used earlier [20,23,29], these have generally been hierarchical, graph theoretic or set theoretic in nature, that have taken into consideration the pairwise scores returned by alignment techniques. The corresponding clustering algorithms generally have high-computational complexity of the order of $O(n^3)$ where n is the number of sequences. In contrast, with extracted numerical features, lower complexity clustering algorithms become applicable.

In this article we propose another feature extraction scheme that overcomes some of the limitations of [30,31]. The number of extracted features is limited to 20, and a weighting scheme in the lines of profile analysis [16,18] is adopted. This helps in incorporating, to a large extent, the positional information of the individual amino acids. Once the features are extracted, a fuzzy genetic clustering strategy [24] is adopted to first evolve a set of prototypes for each superfamily under consideration. Note that hybridization of fuzzy systems and genetic algorithms (GAs) have gained widespread interest in recent years [7].

In clustering (also known as exploratory data analysis) [9,10,12,15,17,28], a set of patterns, usually vectors in a multi-dimensional space, are organized into coherent and contrasted groups, such that patterns in the same group are similar in some sense and patterns in different groups are dissimilar in the same sense. The aim of any clustering technique is to evolve a partition matrix $U(X)$ of the given data set X (consisting of, say, n patterns, $X = \{x_1, x_2, \ldots, x_n\}$) into a number, say c, of clusters $(C_1, C_2, \ldots, C_c)$ such that some measure of goodness of the clusters is maximized. The partition matrix $U(X)$ of size $c \times n$ may be represented as $U = [u_{ik}]$, $i = 1, \ldots, c$ and $k = 1, \ldots, n$, where u_{ik} is the membership of pattern x_k to clusters C_i. In fuzzy partitioning of the data, the following conditions hold:

$$
\begin{aligned}
&0 < \sum_{k=1}^{n} u_{ik} < n \quad && \text{for } i = 1, \ldots, c, \\
&\sum_{i=1}^{c} u_{ik} = 1 && \text{for } k = 1, \ldots, n \text{ and,} \\
&\sum_{i=1}^{c} \sum_{k=1}^{n} u_{ik} = n. &&
\end{aligned}
$$

Fuzzy c-Means (FCM) [6] is a widely used technique that uses the principles of fuzzy sets to partition a data set into a fixed number, c, of clusters; thereby providing the appropriate $c \times n$ partition matrix such that the above conditions hold good. However, FCM has two major limitations: it requires the a priori specification of the number of clusters, and it often gets stuck at suboptimal solutions based on the initial configuration of the system.

The searching capability of GAs has been exploited in [24] to evolve automatically the fuzzy partitions of set of prototypes representing each class. GAs [8,14,25] are randomized search and optimization techniques guided by the principles of evolution and natural genetics. They perform search in complex, large and multimodal landscapes, and provide near optimal solutions for objective or fitness function of an optimization problem. The characteristic of the clustering method [3] is that it does not require the a priori information about the number of clusters, and is also able to come out of locally optimal configurations.

After application of fuzzy clustering, a number of centroids are evolved corresponding to each superfamily. Finally, the nearest neighbor (NN) rule [4,28] is used to classify a set of unknown/query sequences into particular superfamily classes, based on their proximity to the prototypes evolved using the genetic fuzzy clustering technique. This results in a significant improvement in the time required for classifying unknown sequences over the time required for a full NN search where the entire training data is used.

The proposed technique, in fact, overcomes two standard limitations of the NN rule, viz., a requirement to store all the training patterns, and to compute the distances of the query sequence to all the training patterns.

Results for three superfamilies, namely globin, trypsin and ras, demonstrate the effectiveness of the proposed technique with respect to the case where all the training sequences are considered for classification using the same set of features. Comparison with the well-known technique BLAST also shows that the proposed method provides a significant improvement in terms of the time required for classification while providing comparable classification performance.

2. An overview of protein sequence classification

In this section, we briefly describe the representation techniques of the protein sequences and other related issues in the problem of protein classification. Deoxyribose nucleic acid (DNA) is a sequence of four base/nucleotides-Adenine (A), Guanine (G), Thymine (T) and Cytosine (C). DNA is first transcribed into ribose nucleic acid (RNA) which contains Uracil (U) instead of Thymine, while eliminating a lot of junk information.

RNA is thereafter translated into proteins. The translation process considers three consecutive nucleotides, also known as codons. Each codon codes for a unique amino acid. Thus, the consecutive codons translate into a sequence of consecutive amino acids, which essentially is a protein. Proteins are therefore sequences composed of an alphabet of 20 amino acids. Note that there are $4*4*4 = 64$ codons, although there are just 20 amino acids. This is due to the degeneracy of the genetic code. These amino acids are represented by the following set:

$$Am = \{\mathrm{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y}\}.$$

A protein sequence can be of any length and the amino acids can combine in any order. This gives us an idea of the huge number of possible protein structures [21].

As observed, groups of proteins have similarity in functions and structures, and we refer to a group of proteins that share such similarity as a superfamily. An important issue in protein sequence classification is how to encode the protein sequences, i.e., how to represent the protein sequences in terms of feature vectors. Evidently, a good input representation (extraction of features) is crucial for the proper classification of the sequences into superfamilies.

3. Proposed technique of extraction of features

In the proposed encoding scheme, the evolutionary profile information from multiple sequences belonging to a particular superfamily is taken into account. Evolutionary similarity among proteins can be explained as follows. Fig. 1 shows the primary structure of four related proteins. Only a small piece of each protein is shown. By taking a closer look at the structures, the history of evolution in this protein family can be identified. Possibly, the ancestor of the four proteins in Fig. 1 looked like the protein in Fig. 2.

In general, proteins have evolved over time such that although a set of sequences share the same ancestor, structure differences become evident among them because of the evolutionary process. The

LSALSDLHAHKLRVDPVN
LLALSDLHHHKLRVIMVN
LSALSALHHAKLRPIMVN
ASALSDAIAHMIRVDMVI

Fig. 1. Primary structure of four related proteins.

LSALSDLHIHKLRVDMVN

Fig. 2. A possible common ancestor.

differences arise due to some biological changes in form of insertions, deletions and substitutions during cell reproduction. Through a process called *mitosis*, the cell makes a copy of itself and then splits into two daughter cells. Most of the time a protein of the parent cell is exactly duplicated in the daughter cell. However, over long periods of time, errors occur in this copying process. When this happens, a protein in the daughter cell becomes slightly different from the parent. It also happens that these proteins suffer similar degradation in their structure, although a generality of structure is still maintained. As a result of these errors, proteins which share a common ancestor are not exactly alike. However, they inherit many similarities in primary structure from their ancestors. This is known as conservation of primary structure in a protein family. These structural similarities made it possible to create a statistical model of a protein family [16,18].

3.1. *Statistical profile*

In order to construct a statistical profile of a set of sequences, a $20 \times l_{\max}$ probability matrix is computed where $l_{\max}$ = maximum length of a sequence belonging to a particular superfamily. For example, $l_{\max} = 171$ for the sequences of the protein superfamily globin that have been considered in this article. Here, the value at position (i, j) indicates the probability of occurrence of the ith amino acid in position j of the sequence. As an example, consider the model shown in Table 1 which is a simplified statistical profile of the five sequences shown in the Fig. 3. According to this profile, the probability of L in position 1 is 0.8, the probability of A in position 2 is 0.6, and so forth. The probabilities are calculated from the observed frequencies of amino acids in the family of protein sequences (e.g., L occurs in the first position us four out of five sequences, or $\frac{4}{5} = 0.8$).

3.2. *Input feature extraction*

Given a profile, the position-specific weight of any amino acid in a given sequence can be obtained by adding the occurrences of the amino acid at a particular place and the respective probability of the occurrence of that amino acid in that place for the entire family. For example, using this method and the probability matrix shown in Table 1, for the sequence LAADT the weights of the amino acids are

$$Weight(L) = 1 \times 0.8 = 0.8.$$
$$Weight(A) = 1 \times 0.6 + 1 \times 0.8 = 1.4$$

Table 1
Statistical model of five related proteins shown in Fig. 3

Positions	1	2	3	4	5
Prob (L)	0.8	0.4	0.0	0.0	0.0
Prob (A)	0.2	0.6	0.8	0.0	0.0
Prob (H)	0.0	0.0	0.2	0.0	0.0
Prob (D)	0.0	0.0	0.0	0.6	0.2
Prob (T)	0.0	0.0	0.0	0.4	0.0
Prob (R)	0.0	0.0	0.0	0.0	0.6
Prob (V)	0.0	0.0	0.0	0.0	0.2

Sequence1	L	A	A	T	R
Sequence2	L	A	H	D	V
Sequence3	A	L	A	D	R
Sequence4	L	A	A	T	R
Sequence5	L	L	A	D	D

Fig. 3. Five related proteins.

$$Weight(D) = 1 \times 0.6 = 0.6.$$
$$Weight(T) = 1 \times 0 = 0.$$

The weights of all other amino acids are zero. This is because either they appear in irrelevant positions with respect to the already known superfamily members which form our training set or they do not appear at all. So for the sequence LAADT the feature vector is [1.2 0 0.6 0 0 0 0 0 0 0.8 0 0 0 0 0 0 0 0 0 0] representing the weights of features [A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y].

As is evident, the number of features for any sequence will always be equal to 20. In contrast to the method in [30], where a large number of features are neglected in order to reduce the dimensionality, here the loss of information as a result of neglecting some possible input features is absent. Also position specific information is incorporated in the scheme. In contrast to alignment-based methods like BLAST, here each sequence is represented by a set of features, which thereafter allows for the application of several low complexity clustering and classification algorithms.

4. The fuzzy clustering technique

In this section, we describe the use of variable string length genetic algorithms (VGAs) to automatically evolve a set of prototypes representing each class [3]. As in conventional GAs [14], the basic operations in VGAs also are selection, crossover and mutation. The technique is described below in detail.

String representation and population initialization: In *VGA-clustering*, the chromosomes are made up of real numbers (representing the coordinates of the centers). If chromosome i encodes the centers of K_i clusters in N-dimensional space, $K_i \geqslant 2$, then its length l_i is taken to be $N * K_i$.

Each string i in the population initially encodes the centers of a number, K_i, of clusters, where K_i is given by $K_i = rand() \bmod K^*$. Here, $rand()$ is a function returning an integer, and K^* is a soft estimate of the upper bound of the number of clusters. Note that K^* is used only for the generation of the initial population. The actual number of clusters in the data set is not related to K^*, and may be any number greater than, equal to or less than K^*. The K_i centers encoded in a chromosome are randomly selected points from the data set. As an example, suppose a chromosome encodes three cluster centers in two-dimensional space. Suppose the three patterns randomly selected from the data set are (3.2, 4.5), (14.3, 9.8) and (7.2, 10.4). Then the chromosome will look as follows:

$$\text{Chrom1} : (3.2,\ 4.5)\ (14.3,\ 9.8)\ (7.2,\ 10.4).$$

Another chromosome may be present in the same population which encodes, say, five centers, namely, (2.5, 3.2), (10.7, 7.8), (11.2, 5.5), (3.7, 6.2) and (7.5, 5.7). Then the corresponding chromosome will look as follows:

$$\text{Chrom2} : (2.5,\ 3.2)\ (10.7,\ 7.8)\ (11.2,\ 5.5)\ (3.7,\ 6.2)\ (7.5,\ 5.7).$$

Fitness computation: Cluster validity index is often used to indicate the goodness of a partitioning of a data set as obtained by a clustering algorithm. In this article we use one such cluster validity index, the Xie–Beni (XB) index [33] for this purpose. For a data set $X = \{x_1, x_2, \dots, x_n\}$, set of c cluster centers $V = \{v_1, v_2, \dots, v_c\}$, and the corresponding partition matrix U, the XB index is defined as the ratio of the total variation σ to the minimum separation *sep* of the clusters, where σ and *sep* can be written as

$$\sigma(U, V; X) = \sum_{i=1}^{c} \sum_{k=1}^{n} u_{ik}^2 ||x_k - v_i||^2 \tag{1}$$

and

$$sep(V) = \min_{i \neq j}\{||v_i - v_j||^2\}. \tag{2}$$

The XB index is then written as

$$XB(U, V; X) = \frac{\sigma(U, V; X)}{n\, sep(V)} = \frac{\sum_{i=1}^{c}(\sum_{k=1}^{n} u_{ik}^2 ||x_k - v_i||^2}{n\,(\min_{i \neq j}\{||v_i - v_j||^2\})}. \tag{3}$$

Note that when the partitioning is compact and good, value of σ should be low while *sep* should be high. Therefore, the XB index should have a low value when the data has been appropriately clustered. Or, in other words, if the XB index of a particular tuple (U_1, V_1) is XB_1, and that of another tuple (U_2, V_2) is XB_2, and if $XB_1 < XB_2$, then the partitioning corresponding to (U_1, V_1) is taken to be better than that of (U_2, V_2). The objective is to minimize the XB index for achieving proper clustering. The fitness function for chromosome j is defined as $1/XB_j$, where XB_j is the XB index computed for this chromosome. Note that maximization of the fitness function will ensure minimization of the XB index.

Selection: Conventional proportional selection is applied on the population of strings. Here, a string receives a number of copies that is proportional to its fitness in the population. We have used the roulette wheel strategy for implementing the proportional selection scheme.

Crossover: For the purpose of crossover, the cluster centers are considered to be indivisible, i.e., the crossover points can only lie in between two clusters centers. The crossover operator, applied stochastically with probability μ_c, must ensure that information exchange takes place in such a way that both the offspring encodes the centers of at least two clusters. For this, the operator is defined as follows:

Let parents P_1 and P_2 encode K_1 and K_2 cluster centers respectively. $\mathcal{C}_1$, the crossover point in P_1, is generated as $\mathcal{C}_1 = \mathit{rand}()\text{mod}\, K_1$. As before, *rand*() is a function that returns an integer. Let $\mathcal{C}_2$ be the crossover point in P_2, and it may vary in between $[LB(\mathcal{C}_2), UB(\mathcal{C}_2)]$, where $LB()$ and $UB()$ indicate the lower and upper bounds of the range of $\mathcal{C}_2$, respectively. $LB(\mathcal{C}_2)$ and $UB(\mathcal{C}_2)$ are given by

$$LB(\mathcal{C}_2) = \min[2, \max[0, 2 - (K_1 - \mathcal{C}_1)]] \tag{4}$$

and

$$UB(\mathcal{C}_2) = [K_2 - \max[0, 2 - \mathcal{C}_1)]]. \tag{5}$$

Therefore $\mathcal{C}_2$ is given by

$$\mathcal{C}_2 = LB(\mathcal{C}_2) + \mathit{rand}()\text{mod}(UB(\mathcal{C}_2) - LB(\mathcal{C}_2)).$$

As an example, suppose the two chromosomes selected for crossover be

$$\text{Chrom1} : (3.2,\ 4.5)\ (14.3,\ 9.8)\ (7.2,\ 10.4)$$

and

$$\text{Chrom2} : (2.5,\ 3.2)\ (10.7,\ 7.8)\ (11.2,\ 5.5)\ (3.7,\ 6.2)\ (7.5,\ 5.7).$$

Here $K_1 = 3$ and $K_2 = 5$. Let $\mathcal{C}_1 = 2$. Therefore,

$$LB(\mathcal{C}_2) = \min[2, \max[0, 2 - (3 - 2)]] = \min[2, \max[0, 1]] = 1 \tag{6}$$

and

$$UB(\mathcal{C}_2) = [5 - \max[0, 2 - 2)]] = 5. \tag{7}$$

Therefore

$$\mathcal{C}_2 = 1 + \mathit{rand}()\text{mod}(5 - 1).$$

Let $\mathcal{C}_2 = 3$. Therefore after crossover, the two offspring will be

$$\text{Offspring1} : (3.2,\ 4.5)\ (14.3,\ 9.8)\ (3.7,\ 6.2)\ (7.5,\ 5.7)$$

and

$$\text{Offspring2} : (2.5,\ 3.2)\ (10.7,\ 7.8)\ (11.2,\ 5.5)\ (7.2,\ 10.4).$$

Mutation: Each chromosome undergoes mutation with a fixed probability μ_{m}. Since floating point representation is considered in this article, we use the following mutation. A number δ in the range [0, 1] is generated with uniform distribution. If the value at a gene position is v, after mutation it becomes $(1 \pm 2 * \delta) * v$, when $v \neq 0$, and $\pm 2 * \delta$, when $v = 0$. The '+' or '−' sign occurs with equal probability.

As an example, suppose Offspring1 undergoes mutation in the first dimension of the second gene position (cluster center). After mutation with $\delta = 0.2$, it may look as follows:

$$(3.2,\ 4.5)\ ((1 - 2 * 0.2) * 14.3,\ 9.8)\ (3.7,\ 6.2)\ (7.5,\ 5.7),$$

which is

$$(3.2,\ 4.5)\ (8.58,\ 9.8)\ (3.7,\ 6.2)\ (7.5,\ 5.7).$$

5. The NN classification

In this section the nearest neighbor (NN) classification procedure in briefly described. Let us consider a set of n labeled pattern $\{x_1, x_2, \ldots, x_n\}$ belonging to one of the classes $C_1, C_2, \ldots, C_c$. The NN classification rule assigns an unlabeled pattern y to the class of its (NN) $x_i \in \{x_1, x_2, \ldots, x_n\}$ such that

$$D(x_i, y) = \min_l \{D(x_l, y)\}, \quad l = 1, 2, \ldots, n, \tag{8}$$

where D is any distance measure defined over the feature space. Since the aforesaid scheme employs the class label of only the NN to y, this is known as the NN rule. As an example consider six data points labeled into two classes as follows:

Class 1 : (9, 10), (10, 10) and (10, 9),
Class 2 : (14, 13), (13, 14) and (13, 13).

Suppose the unknown point is (11,11). Then after computing its distance from all the six labeled points, it can be seen that the NN is the point (10,10) that belongs to Class 1. Hence the unknown point is labeled as Class 1.

The details of the NN rule along with the probability of error are available in [10,11,13 ,28].

The NN rule, though conceptually very simple and appealing, has two severe limitations. These are that all the n training patterns need to be stored, and n distance computations need to be carried out for classifying an unknown sequence. These limitations are overcome in this article by the application of fuzzy clustering on the training patterns. Using the variable length genetic fuzzy clustering technique described in the previous section, the training data corresponding to each superfamily class is first partitioned into a number of clusters that is evolved automatically. Thereafter, instead of the full training data, only the prototypes for each superfamily class (or, the centers of the clusters evolved by the genetic fuzzy clustering technique) are used as the training data. The NN rule is then applied with the evolved set of prototypes (whose size is much smaller than the size of the entire data), rather than the full data set. This is expected to result in a significant improvement in the computation time.

6. Results

Experiments were carried out to evaluate the effectiveness of the proposed scheme. The data used in the experiments were obtained from the International Protein Sequence Data base, release 75, in the Protein Information Resource (PIR) maintained by the National Biomedical Research Foundation (NBREF-PIR) at the Georgetown University Medical Center. Three superfamilies, viz., globin, trypsin and ras were considered. We have taken 500 sequences from each of these superfamilies. The maximum variation in lengths of the sequences for these superfamilies was of about 70 residues. Each class was divided into training and test sets comprising 250 sequences each. Results were compared with respect to both the classification performance and the timing requirements when executing on SUN-BLADE with a processor speed of 1.2 GHz. For the purpose of comparison, experiments were carried out using BLAST (obtained from the http://www.ncbi.nlm.nih.gov site), NN rule using the full training data (instead of evolved prototypes), and also with prototypes evolved using the well-known fuzzy clustering algorithm, the fuzzy c-means (FCM) algorithm [6] (with the same number of clusters as obtained using the genetic

Table 2
Confusion matrix obtained using BLAST

	Globin	Trypsin	Ras	% Score
Globin	205	45	0	82.0
Trypsin	77	168	5	67.2
Ras	0	0	250	100.0
Overall				83.06

Table 3
Confusion matrix obtained using full NN rule

	Globin	Trypsin	Ras	% Score
Globin	204	30	16	81.6
Trypsin	0	217	33	86.8
Ras	0	53	197	78.8
Overall				82.4

Table 4
Confusion matrix obtained using the prototypes evolved using genetic fuzzy clustering

	Globin	Trypsin	Ras	% Score
Globin	186	37	27	74.4
Trypsin	4	186	69	74.4
Ras	0	9	241	96.4
Overall				81.73

fuzzy clustering method). For the genetic fuzzy clustering, population size was taken to be 50, and the probabilities of crossover and mutation were fixed at 0.8 and 0.1, respectively. A maximum of 100 iterations was executed. The number of prototypes evolved for globin, trypsin and ras were 2, 15 and 16, respectively.

The confusion matrix obtained using BLAST is shown in Table 2. The confusion matrix corresponds to the actual classes along the rows, and the computed classes along the columns. The entry (i, j) in the confusion matrix shows the number of points that actually belongs to a particular class, say i, but is classified to class j. The class wise and overall correct recognition scores are also shown in the table. The corresponding tables for the full NN rule (using the full data for training), the proposed genetic fuzzy clustering algorithms and FCM are shown in Tables 3–5, respectively. It is found that the performance of BLAST and the full NN method are comparable in terms of percentage recognition scores. The score obtained by the using the prototypes evolved using the genetic fuzzy clustering scheme is also reasonably good, while that obtained using FCM are quite poor. This suggests the effectiveness in our examples. When timing comparisons are made, the proposed method and FCM outperform the full NN rule by about 20 times, while the timing requirement of BLAST is the largest at 6 min (Table 6). This demonstrates that the proposed method provides a reasonably good result significantly faster than both BLAST and the full NN rule (Table 6).

Table 5
Confusion matrix obtained using the prototypes evolved using FCM

	Globin	Trypsin	Ras	% Score
Globin	180	31	39	72.0
Trypsin	33	182	35	72.8
Ras	0	70	180	72.0
Overall				72.26

Table 6
Time required for classification

Method	Time required
BLAST	6 min
Full NN rule	2.42 s
Proposed method	0.12 s
FCM-based method	0.12 s

7. Conclusion

In this article, a hybrid scheme for protein superfamily classification is proposed that combines a technique for feature extraction, fuzzy clustering and NN classification. The feature extraction scheme incorporates some amount of positional information of the amino acids in the sequences. The genetic fuzzy clustering method is able to generate a good set of prototypes representing each class, and the NN rule is used for classification into the appropriate class. Results are reported on three superfamily classes namely, globin, trypsin and ras. It is found that although the classification result obtained by BLAST is the best, the one based on fuzzy clustering and NN classification is also reasonably good, while the timing requirement is significantly lower as compared to BLAST as well as the full NN approach.

There are several ways in which this work can be extended further. First of all, if a pre-alignment is made before computing the probability matrix, then this may result in a better weighting scheme, and hence better extracted features. Again, other feature extraction schemes also need to be developed. The fuzzy clustering adopted in this article uses the XB index as the optimizing criterion. Other indices like [26] could be applied and a comparative study needs to be performed. Note that these indices are based on the compactness of the classes in the data set. Validity indices that evaluate different characteristics, other than compactness, need to be studied in this context in future. New indices may have to be developed for this purpose. Finally, the effectiveness of other classification methods, including neural network-based classifiers, for superfamily classification needs to be studied.

References

[1] S.F. Altschul, T.L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller, D.J. Lipman, Gapped BLAST and PSI-BLAST: a new generation of protein database search programs, Nucleic Acids Res. 25 (1997) 3389–3402.

[2] A.N. Arslan, O. Egecioglu, P.A. Pevzner, A new approach to sequence comparison: normalized local alignment, Bioinformatics 17 (4) (2001) 327–337.

[3] S. Bandyopadhyay, U. Maulik, Non-parametric genetic clustering: comparison of validity indices, IEEE Trans. Systems Man Cybernet. Part-C 31 (1) (2001) 120–125.
[4] S. Bandyopadhyay, U. Maulik, Efficient prototype reordering in nearest neighbor classification, Pattern Recognition 35 (2002) 2791–2799.
[5] A. Baxevanis, F.B.F. Ouellette (Eds.), Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins, Wiley, New York, 1998.
[6] J.C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum, New York, 1981.
[7] O. Cordon, F. Gomide, F. Herrera, F. Hoffmann, L. Magdalena, Ten years of genetic fuzzy systems: current framework and new trends, Fuzzy Sets and Systems 141 (2004) 5–31.
[8] L. Davis (Ed.), Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, 1991.
[9] D. Defays, An efficient algorithm for a complete link method, Comput. J. 20 (1977) 364–366.
[10] P.A. Devijver, J. Kittler, Pattern Recognition: A Statistical Approach, Prentice-Hall, London, 1982.
[11] R.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.
[12] B.S. Everitt, Cluster Analysis, third ed., Halsted Press, New York, 1993.
[13] K. Fukunaga, Introduction to Statistical Pattern Recognition, Academic Press, New York, 1990.
[14] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, New York, 1989.
[15] J.A. Hartigan, Clustering Algorithms, Wiley, New York, 1975.
[16] R. Hughey, A. Krogh, Hidden markov models and sequence analysis: extension and analysis of the basic method, Comput. Appl. Biosci. 12 (2) (1996) 95–107.
[17] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Prentice-Hall, Englewood Cliffs, NJ, 1988.
[18] R. Karchin, R. Hughey, Weighting hidden markov models for maximum discrimination, Bioinformatics 14 (9) (1998) 772–782.
[19] T. Koski, Hidden Markov Models for Bioinformatics, Kluwer Academic Publishers, Dordrecht, 2001.
[20] A. Krause, M. Vingron, A set-theoretic approach to database searching and clustering, Bioinformatics 14 (5) (1998) 430–438.
[21] N.M. Luscombe, D. Greenbaum, M. Gerstein, What is bioinformatics: a proposed definition and overview of the field, Methods Inform. Med. 40 (2001) 346–358.
[22] B. Ma, J. Tromp, M. Li, Pattern hunter: faster and more sensitive homology search, Bioinformatics 18(3) (2002) 440–445.
[23] H. Matsuda, T. Ishihara, A. Hashimoto, A clustering method for molecular sequences based on pairwise similarity, in: Proceedings of the Seventh Workshop on Genome Informatics, Tokyo, Universal Academy Press, 1996.
[24] U. Maulik, S. Bandyopadhyay, Fuzzy partitioning using a real-coded variable-length genetic algorithm for pixel classification, IEEE Trans. Geosci. Remote Sensing 41 (5) (2003) 1075–1081.
[25] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer, New York, 1992.
[26] M.K. Pakhira, S. Bandyopadhyay, U. Maulik, Validity index for crisp and fuzzy clusters, Pattern Recognition 37 (3) (2004) 487–501.
[27] J.C. Setubal, J. Meidanis, Introduction to Computational Molecular Biology, PWS Publishing Company, Boston, 1997.
[28] J.T. Tou, R.C. Gonzalez, Pattern Recognition Principles, Addison-Wesley, Reading, MA, 1974.
[29] P.A. Vijaya, M.N. Murty, D.K. Subramanian, Supervised *k*-medians algorithm for prototype selection for protein sequence classification, in: Proceedings of the International Conference on Advances in Pattern Recognition, Allied Press, India, 2003, pp. 129–132.
[30] J.T.L. Wang, Q.C. Ma, D. Shasha, C.H. Wu, Application of neural networks to biological data mining: a case study in protein sequence classification, in: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2000), Boston, MA, 2000, pp. 305–309.
[31] J.T.L. Wang, Q.C. Ma, D. Shasha, C.H. Wu, New techniques for extracting features from protein sequences, IBM Systems J. 40 (2) (2001) 426–441 (Special Issue on Deep Comput. Life Sci.).
[32] C.H. Wu, J.W. McLarty, Neural Networks and Genome Informatics, Elsevier, Amsterdam, 2000.
[33] X.L. Xie, G. Beni, A validity measure for fuzzy clustering, IEEE Trans. Pattern Anal. Mach. Intell. 13 (1991) 841–847.