Hindawi Publishing Corporation
Computational Biology Journal
Volume 2013, Article ID 898090, 12 pages
http://dx.doi.org/10.1155/2013/898090

Research Article

Two-Stage Approach for Protein Superfamily Classification

Swati Vipsita and Santanu Ku. Rath

Department of Computer Science & Engineering, N.I.T., Rourkela, Odisha 769008, India

Correspondence should be addressed to Swati Vipsita; vipsita.nitrkl@gmail.com

Received 26 February 2013; Revised 3 June 2013; Accepted 3 June 2013

Academic Editor: Qianzhong Li

We deal with the problem of protein superfamily classification in which the family membership of newly discovered amino acid sequence is predicted. Correct prediction is a matter of great concern for the researchers and drug analyst which helps them in discovery of new drugs. As this problem falls broadly under the category of pattern classification problem, we have made all efforts to optimize feature extraction in the first stage and classifier design in the second stage with an overall objective to maximize the performance accuracy of the classifier. In the feature extraction phase, Genetic Algorithm- (GA-) based wrapper approach is used to select few eigenvectors from the principal component analysis (PCA) space which are encoded as binary strings in the chromosome. On the basis of position of 1's in the chromosome, the eigenvectors are selected to build the transformation matrix which then maps the original high-dimension feature space to lower dimension feature space. Using PCA-NSGA-II (non-dominated sorting GA), the nondominated solutions obtained from the Pareto front solve the trade-off problem by compromising between the number of eigenvectors selected and the accuracy obtained by the classifier. In the second stage, recursive orthogonal least square algorithm (ROLSA) is used for training radial basis function network (RBFN) to select optimal number of hidden centres as well as update the output layer weighting matrix. This approach can be applied to large data set with much lower requirements of computer memory. Thus, very small architectures having few number of hidden centres are obtained showing higher level of performance accuracy.

1. Introduction

Protein superfamily classification deals with the prediction of family membership of newly discovered proteins. This helps the drug analyst for discovery of new drugs required in the treatment of new diseases. This classification helps in predicting the protein function and/or structure of the unknown sequence, thus avoiding the expensive biological (wet) experiments in the laboratory. Once a particular sequence S causing disease D is classified to a superfamily F_i, the researchers can design some new drugs by trying some combination of existing drugs for family F_i. Thus, this classification problem helps the researchers in treatment of diseases by discovering new drugs [1]. Among the major techniques developed in the past, the popular BLAST tool [19] represents the simplest nearest neighbour approach and exploits pairwise local alignments to measure sequence similarity. Another type of direct modelling method is based on hidden markov models (HMMs) [2], (Karplus et al., 1998). After constructing an HMM for each family, protein queries can be easily scored against all established HMMs. This can be done by calculating the log-likelihood of each model for the unknown sequence and then selecting the class label of the most likely model. The Motif Alignment and Search Tool (MAST) (Bailey and Gribskov, 1998) is based on the combination of multiple motif-based statistical score values. According to this scheme, groups of probabilistic motifs discovered by the MEME (Multiple Em for Motif Elicitation) algorithm (Bailey and Elkan, 1994) are used to construct protein profiles for the families of interest.

As there is an exponential growth in size of biological database, computational intelligent techniques possess the real challenge to handle this huge database and retrieve valuable knowledge from it. These intelligent techniques are mostly applied for feature selection, dimensionality reduction as well as in the design and optimization of the classifier. Feature extraction and feature selection have a great impact on the performance accuracy of the classifier. Singular value decomposition (SVD) for n-gram feature reduction and feedforward neural network (FFNN) trained using backpropagation (BP) algorithm are implemented in [3]. PCA is used

to reduce the size of n-gram feature vector in [4]. Besides the global features, a linear correlation coefficient (LCC) is evaluated as well as local features are extracted from motif content, and Bayesian neural network (BNN) as a classifier is implemented in [5]. A generalized radial basis function (GRBF) neural network architecture that generates fuzzy classification rules is used for protein sequence classification in [6], and comprehensibility of fuzzy rule-based classifier with comparable classification accuracy is shown in [7]. Principal component null space analysis (PCNSA) linear classifier in [8] reported excellent results compared to those of neural networks and support vector machines (SVM) which efficiently handled high-dimensional data thereby maximizing accuracy to much extent. Extreme learning machine (ELM) needed up to four orders of magnitude less training time compared to BP network was used to classify protein sequences with ten classes of superfamilies downloaded from a public domain database in [9]. Feature selection based on low relative entropy values using SVM as classifier is shown in [10]. Feature selection based on feature ranking method is implemented in [11] where a novel steady-state genetic algorithm for extracting fuzzy classification rules from data is used to generate a compact set of interpretable fuzzy rules that is used for protein superfamily classification. An implementation of subtractive clustering in standard RBF and modular RBF is shown in [12]. Based on probabilities of occurrences of the amino acids in the different positions of the sequences, a genetic fuzzy clustering approach evolves a set of prototypes representing each class. The nearest neighbour rule is then used to classify an unknown sequence into a particular class, based on its proximity to the prototypes [13]. In [14, 15], the classification problem is mapped as a multiobjective problem (MOP) having two objectives such as minimization of number of features selected and minimization of misclassification error rate which are solved using the weighted sum approach.

This paper provides insight to solve the protein superfamily classification problem in two major stages. In the first stage, we have implemented PCA-NSGA-II, a hybridized approach which tries to solve the trade-off problem in selecting optimal number of significant eigenvectors by maintaining a good level of performance accuracy of the inductive algorithm. The encoding of chromosome becomes a very difficult task as feature vector extracted from amino acid sequence is too high. So to overcome this problem, eigenvectors having nonzero eigenvalues are encoded in the chromosome, and GA helps in searching the eigenspace to select the distinguishing eigenvectors. Probabilistic neural network (PNN) is used as an inductive algorithm, and the evaluation function used in this wrapper approach is the minimization of the misclassification error rate of the PNN over the test samples. (A detailed description of wrapper approach and inductive algorithm is discussed in [16].) The major limitations of not using PNN as a classifier are that all the training patterns need to be stored for classifying a new pattern. As the size of the training set increases, the network becomes too complex as every node in the pattern layer represents a training sample. So, radial basis function network (RBFN) is preferred to be a classifier, but the major issue lies in finding out the exact number of hidden centres as well as the update matrix connecting the hidden and the output layer. Recursive orthogonal least square algorithm (ROLSA) finds the real solution to the above-mentioned problems.

The rest of the paper is organized as follows: Section 2 describes data acquisition and preprocessing steps from amino acid sequence. Section 3 describes in detail the feature extraction from eigenspace using PCA-NSGA-II with relevant algorithms. Section 4 describes the implementation of ROLSA used for training of RBFN. Section 5 describes the experiment details, and Section 6 discusses the results of numerical simulations. Section 7 concludes the paper.

2. Data Acquisition and Preprocessing from Amino Acid Sequence

In sequence-based classification of proteins the data acquisition and data preprocessing stage involves the retrieval of amino acid sequences from the publicly available database such as Uniprot, Pubmed, and Protein Information Resource (PIR). The data are downloaded and saved in FASTA format. The sequence-based classification methods can be divided into three broad categories such as *feature-based classification* [17, 18] where feature vectors are built from long amino acid sequences and these features help in classification of protein to their superfamily. In *sequence distance-based classification* [19], the distance function determines the similarity between sequences, and the quality of classifier can then be measured. The third category is *model-based classification* where hidden markov model (HMM) [2] or some other statistical models are used to classify the sequences [20]. Out of these three categories, we emphasize on feature-based classification from amino acid sequences for protein superfamily classification. The schematic representation of data acquisition and preprocessing is shown in Figure 1.

In general, the genetic code specifies 20 standard amino acids such as

$$\Sigma = (A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y). \quad (1)$$

For protein feature selection, the two gram features such as

$$(AA, AC \cdots AY), (CA, CC \cdots CY), \ldots, (YA, YC \cdots YY) \quad (2)$$

are selected. The total number of possible bigrams from a set of 20 amino acids is 20^2, that is, 400. The two gram features represent the majority of the protein features. Two grams have the advantages of being length invariant, insertion/deletion invariant, not requiring motif finding and allowing classification based on local similarity [3]. Apart from this, bigrams reflecting the pattern of substitution of amino acids are also extracted. For this purpose, equivalence classes of amino acids that substitute for one another are derived from the percent accepted mutation matrix (PAM) [21]. Exchange grams are similar but are based on a many-to-one translation of the amino acid alphabet into a six

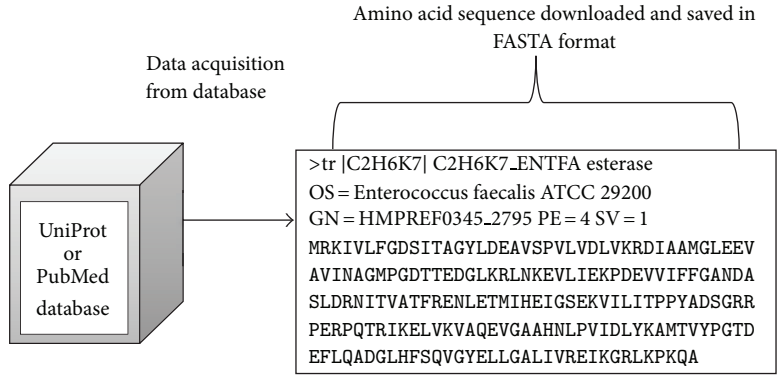

Figure 1: Schematic representation of Data acquisition and data preprocessing.

letter alphabet that represents six groups of amino acids, which represent high evolutionary similarity. Generally the exchange groups used are

$$e1 = \{H, R, K\}, \quad e2 = \{D, E, N, Q\},$$
$$e3 = \{C\}, \quad e4 = \{S, T, P, A, G\}, \quad (3)$$
$$e5 = \{M, I, L, V\}, \quad e6 = \{F, Y, W\}.$$

The exchange groups statistically describe the probability of one amino acid replacing another over time. The total number of possible bigrams on these six substitution groups is 6^2, that is, 36. Thus the overall bigram features extracted compute to 436 values, 400 corresponding to the consecutive pairs of amino acids and 36 corresponding to the consecutive pairs of substitution groups. Besides that, the amino acid distribution (20) and exchange group distribution (6), some other features are also taken into account.

Therefore, for every amino acid sequence, the 470 features were processed to build the fixed dimension feature vector as follows:

$X^{(1)}, X^{(2)}, \ldots, X^{(5)}$ = atomic composition (C, H, N, O, S),

$X^{(6)}$ = molecular weight,

$X^{(7)}$ = isoelectric point,

$X^{(8)}$ = average mass of protein sequence,

$X^{(9)}, X^{(8)}, \ldots, X^{(28)}$ = amino acid distribution,

$X^{(29)}, X^{(30)}, \ldots, X^{(428)}$ = two gram distribution,

$X^{(429)}, X^{(430)}, \ldots, X^{(434)}$ = exchange group distribution,

$X^{(435)}, X^{(434)}, \ldots, X^{(470)}$ = two gram exchange group distribution.

If we assume total "m" number of instances, then the matrix size becomes $m \times 470$ which is a matrix having large number of sparse entries.

3. Feature Extraction Using PCA-NSGA II

3.1. Dimension Reduction of Long Feature Vector. Jain et al. in [22] had clearly distinguished the concept of feature selection from feature extraction. Feature selection refers to some algorithm or technique which selects best subset of features from the original feature set. But feature extraction refers to some techniques which performs some transformation of the original feature matrix to create new feature matrix by discarding the features having low discrimination ability. In other words, feature extraction can be described as, given an $n \times d$ pattern matrix A (n points in a d-dimensional space), an $n \times m$ pattern matrix B is derived, such that $m \ll d$ where $B = AH$ and H is a $d \times m$ transformation matrix.

3.2. Feature Extraction Using Principal Component Analysis (PCA). The concept of PCA was developed by Karl Pearson in 1901. Principal component analysis (PCA) is a statistical technique used to transform a data space of high dimension into a feature space of lower dimension having the most significant features. PCA rigidly rotates the axes of the p-dimensional space to new positions (principal axes) such that principal axis 1 has the highest variance, axis 2 has the next highest variance, and so on. The covariance among each pair of the principal axes is zero so the principal axes are uncorrelated.

3.2.1. Basic Steps of PCA

(i) Calculate mean as the average value across each dimension and then subtract the mean value from each of the data dimension thereby producing data set with zero mean.

(ii) Compute the covariance matrix as $S = A * A^T$ and find out the eigenvalues.

(iii) Sort the eigenvalues in a decreasing order and let them be denoted as $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_M$. Let the corresponding eigenvectors be denoted as $a_1 a_2, \ldots, a_M$.

(iv) Select the first d eigenvectors from M vectors such as $d \ll M$. The cumulative percentage of variance guides

in selecting the top eigenvectors from the covariance matrix whose threshold value is fixed by the user:

$$\text{cum.\% of var.} = \left(\frac{\text{eigen value for the component}}{\text{total eigen values of the cov. matrix}}\right) * 100\%. \tag{4}$$

(v) Finally, based on selected eigenvectors, project the data set into lower dimension as given by

$$G \longleftarrow [a_1 a_2, \ldots, a_d], \quad \text{where } d \ll M. \tag{5}$$

If x is a test point

$$x \in R^M \longrightarrow xG \in R^d. \tag{6}$$

3.3. Multiobjective Genetic Algorithm (MOGA). Most real world problems are complex; in the sense they may be nonlinear, multimodal, and stochastic, and there may be more than one parameter which are needed either to be minimized or to be maximized. These types of problems are referred to as multiobjective problems (MOP) which can be solved by various approaches. A survey of various multiobjective evolutionary techniques is described in [23]. The most simple method among all approaches is to form a composite objective function as the weighted sum of various objectives, and the weight value is assigned as per the priority of individual objective. This technique is otherwise referred to as *preference-based multiobjective optimization* in which a multiobjective problem is mapped as a single objective which is a composite of more than one objective. This can be denoted as

$$F = w_1 f_1 + w_2 f_2 + \cdots + w_n f_n, \tag{7}$$

where $f_1, f_2, \ldots, f_n$ denotes the various objective functions. In the above weighted sum approach, the major practical problem lies in the proper selection of weights. So to handle MOP, pareto optimal solutions are mostly preferred as they can handle the trade-off problem arising due to various objective functions. A MOP should achieve the following three goals [24].

(i) The best known pareto front should be as close as possible to the true pareto front.

(ii) The best known solutions in the pareto front should be uniformly distributed and diverse enough to show the exact concept of trade-off due to conflicting objectives.

(iii) The best known pareto front should capture the whole spectrum of the pareto front. The main objective is to extend the pareto fronts at both ends to explore new extreme solutions.

In 2001, Rudolph introduced the concept of elitism to MOP. Nondominated sorting using NSGA-II was suggested by Deb et al. in [25], which requires time complexity of $O(MN^2)$ where M is the number of objectives and N is the population size. It is shown that NSGA-II was able to maintain a better spread of solutions and converge very near to the true pareto front in comparison to elitist multiobjective evolutionary algorithm such as Pareto archived evolution strategy (PAES) and strength pareto evolutionary algorithm (SPEA). Farina et al., in 2004 [26], have addressed the growing need of solving dynamic MOP in which the pareto optimal set changes with respect to time or with the iteration of the optimization process.

3.4. PCA-NSGA-II. Generally, traditional PCA selects the top few eigenvectors having higher eigenvalues. But Balci and Atalay in [27] have illustrated that eigenvectors having large eigenvalues may not always have a great impact regarding the target concept of interest.

In [28], it is shown that, in face detection and vehicle detection experiments, the eigenvectors having low eigenvalues encode more lighting information and also encode some specific local features. So, GA is implemented which searches the eigenspace to select subset of eigenvectors. The two objectives such as selection of minimum number of significant eigenvectors and minimization of classification error rate are solved using weighted sum approach by giving proper weight values to the objective functions. Zhao et al. in [14] also applied the weighted sum approach for extracting features from motif content and protein content where support vector machine (SVM) is used as an inductive algorithm. Zhao et al. in [15] developed a hybrid GA/RBFNN technique which selects features from protein sequences and trains the RBF neural network simultaneously. The weight factors are assumed to be 40000 and 0.1 for the recognition rate and number of features removed from the original feature set, respectively.

In our approach, PCA is hybridized with the elitist nondominated sorting GA or NSGA-II for feature subset selection from the eigenspace. The encoding of chromosome is done as string of 0's and 1's. The trade-off between two objectives such as minimization of number of eigenvectors selected and minimization of misclassification error rate is solved by generating pareto fronts. The goodness of a chromosome is evaluated based on the number of 1's in the chromosome string. In our problem, GA searches the eigenspace comprising of top 67 eigenvectors having nonzero eigenvalues. The eigenvectors above 67 have very small eigenvalues nearly equal to 0. So the length of chromosome is fixed at 67. After mapping the original feature space to lower dimension using the transformation matrix, the reduced feature matrix is given as input to the probabilistic neural network (PNN). PNN evaluates the second fitness value of chromosome, that is, the misclassification error rate obtained by the selected eigenvectors. The best solutions are derived from the lowest pareto front. The algorithm for implementing PCA-NSGA-II is described below, and the brief overview of the entire process is shown in Figure 2. The NSGA-II procedure is outlined in Algorithm 2. The crowded sorting of the solutions in a front i is evaluated using crowding distance metric described in Algorithm 3. Here, the boundary solutions are assigned large distance values

Let population be denoted as N
Probability of crossover be denoted as P_c
Probability of mutation be denoted as P_m
Fitness function be denoted as $f_1, f_2 \cdots f_n$
Pareto fronts be denoted as $F_1, F_2 \cdots F_m$
$Gen = 0$
repeat
Step 1. $Gen = Gen + 1$.
Step 2. Initialize population P_0.
Initialize N number of chromosomes as random individuals which are encoded as strings of 0's and 1's in the chromosome. The length of chromosome depends on the total number of non-zero eigen vectors having non-zero eigen values. {1 indicates inclusion of the eigen vector in the covariance matrix and 0 represents discard of the eigen vector.}
Step 3. Evaluate fitness function (f_1) = number of 1's in the chromosome string.
Step 4. Evaluate $B = AH$ where A is the original matrix and H is the transformation matrix. {Based on eigen values selected, map the feature matrix to lower dimension by multiplying the original matrix with the transformation matrix.
Step 5. Evaluate fitness f_2 = misclassification error rate of the classifier taking B as input matrix.}
Step 6. Considering f_1 and f_2, perform non-dominated sorting using NSGA-II( ) and generate pareto fronts such as $F_1, F_2, \ldots, F_m$.
Step 7. Calculate the crowding distance of all solution points using the crowding distance( ).
Step 8. Perform tournament selection by selecting N random pairs from P_t.
Step 9. Use the crowded comparison operator( ) to select the most widely spread solutions.
Step 10. Perform pairwise crossover and bitwise mutation to create new offspring.
Step 11. Let the new population be denoted as P_{t+1}.
until ($|F_1| \geq 90\%$ of N)

Algorithm 1: PCA-NSGA-II.

(∞) whereas the intermediate solutions are assigned with crowding distance using the metric. Thus, the population is arranged in descending order of magnitude of the crowding distance values. The crowded tournament selection operator (described in Algorithm 4) decides which solution to enter into the mating pool and become the parents for next generation. The operator randomly chooses two solutions and returns the winner of the tournament based on two conditions. The first condition ensures the solution to be winner if it lies on a better nondominated front. The second condition resolves the tie if both solutions lie on the same front by selecting the solution with a larger value of crowding distance (Algorithm 1).

4. Radial Basis Function Network as a Classifier

4.1. Brief Concept and Architecture of RBFN. A RBF network consists of three layers, namely, the input layer, the hidden layer, and the output layer. The input layer broadcasts the coordinates of the input vector to each of the units in the hidden layer (Figure 3).

The inputs of hidden layer are the linear combinations of scalar weights and the input vector $[x_1, x_2, \ldots, x_n]^T$, where the scalar weights are usually assigned unity values. Each unit in the hidden layer then produces an activation based on the associated radial basis function. The output layer yields a vector $[y_1, y_2, \ldots, y_m]^T$ for m outputs by linear combination of the outputs of the hidden nodes to produce the final output:

$$y = f(x) = \sum_{i=1}^{k} w_i \phi_i(x), \tag{8}$$

where $f(x)$ is the final output, $\phi_i(\cdot)$ denotes the radial basis function of the ith hidden node, w_i denotes the hidden to output weight corresponding to the ith hidden node, and k is the total number of hidden nodes. A normalized Gaussian function is usually used as the radial basis function, that is,

$$\phi_i(x) = \exp\left(-\frac{\|x - c_i\|^2}{2(\sigma_i)^2}\right), \tag{9}$$

where $[x_1, x_2, \ldots, x_n]^T$ denotes the input vector, $[c_{1i}, c_{2i}, \ldots, c_{ni}]^T$ denotes the ith center vector, and $(\sigma_i)^2$ represents the width parameter.

RBFN is an effective tool for pattern classification problem as it has good generalization and approximation ability with a simple network structure. This network has been successfully used in many applications such as pattern classification, system identification, nonlinear function approximation, adaptive control, and speech recognition. RBFN has a faster training procedure in comparison to multilayer perceptron (MLP) trained using backpropagation (BP) algorithm. In generalized RBF, the supervised learning of the center

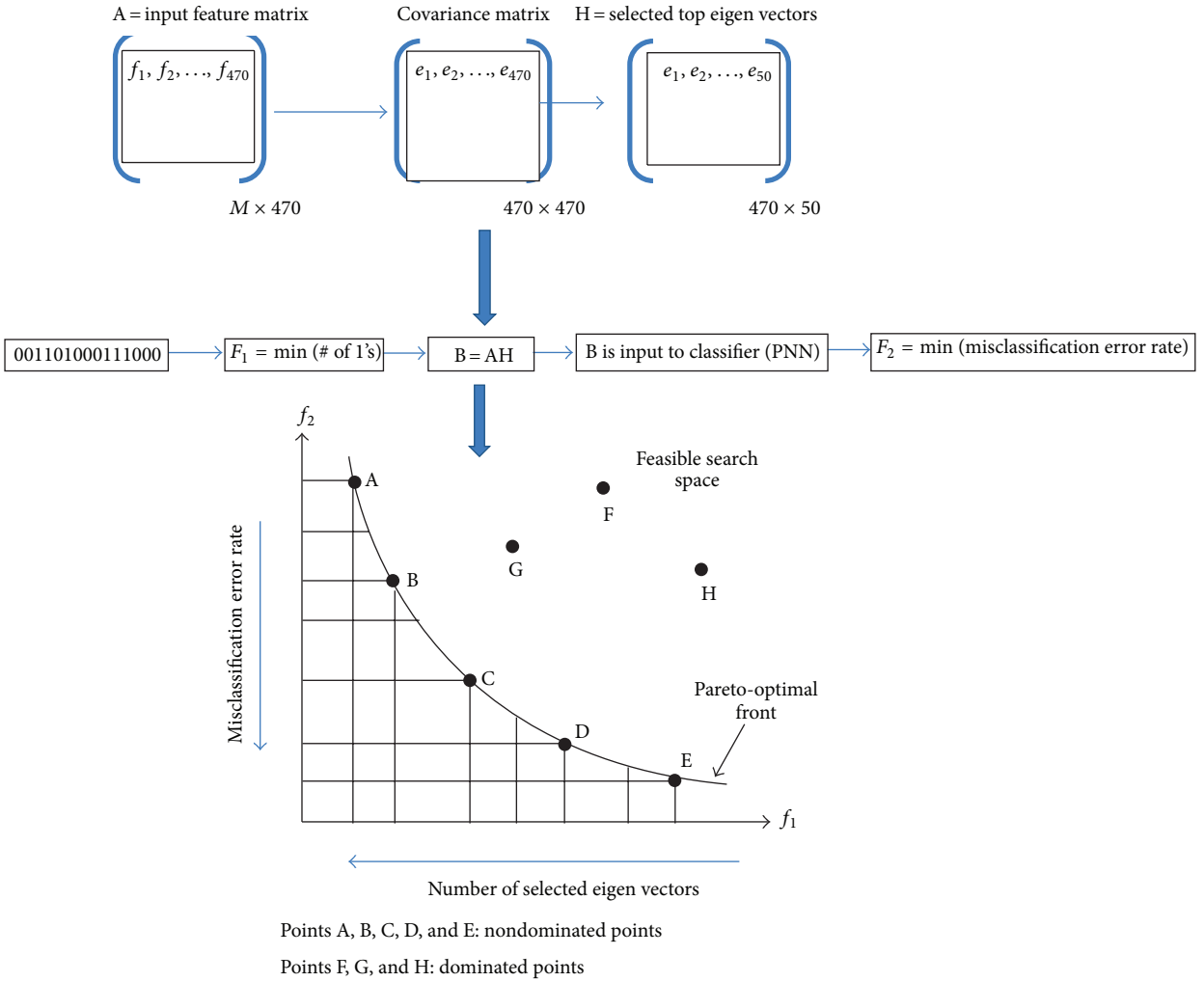

Figure 2: Brief overview of PCA-NSGA-II.

location as well as output layer weights and the Gaussian spread (σ) are performed based on error correction learning rule using a gradient descent procedure. In [29], Neruda and Kudová presented three learning methods of RBFN such as gradient-based learning, three-step unsupervised learning, and evolutionary algorithms, and it was examined on few benchmark problems. It was observed that gradient-based learning performed better in terms of error measured on both the training and test data set. The three-step learning was the fastest but the convergence rate of GA was too high.

In the design and implementation of RBFN, a very critical issue lies in selecting the optimal number of hidden centres as the number of basis function controls the complexity and generalization ability of the network. If too many nodes are selected in the hidden layer, this may include redundant data which results in large network structure. Thus, the computational overhead is too high when we classify an unknown pattern. The network will also have poor generalization capability as it becomes over sensitive to the training data and thereby fails to recognize the noisy samples. On the contrary, very few number of hidden centres in the hidden layer may lower the generalization ability of the trained network. Many researchers used evolutionary optimization techniques to optimize the structure of RBFN.

Implementation of Genetic Algorithm (GA) for structure optimization of RBFN is shown in [30] where each network is coded as a variable length string with distinct integers, and both the single objective and multiobjective functions have been proposed to evaluate network fitness. In 2003, Guo et al. [31] used GA to optimize the parameters of RBFN, and a hybrid learning algorithm further adjusts the parameter values. González et al., in 2003 [32], implemented multiobjective evolutionary algorithm (MOEA) in which global mutation operators based on matrix transformation such as singular value decomposition (SVD) and orthogonal least square (OLS) have been used. Multiobjective structure selection method using MOGA is shown in [33] where the structure of RBFN is encoded as chromosome of GA, and the pareto optimal solutions are obtained from the pareto optimal fronts which solves the trade-off problem between model accuracy and complexity. In [34], GA with hybrid learning algorithm (HLA) has outperformed GA, ROLSA,

Let n_i denotes the domination count i.e the number of solutions which dominates solution i.
S_i denotes set of solutions which solution i dominates.
  Initialize $n_i = 0$ and $S_i = \phi$ for every solution $i \in P$.
  for $(\forall j \neq i)$ *and* $j \in p$ **do**
    if $i \preccurlyeq j$ **then**
      Update $S_p = S_p \cup j$
    else $\{j \preccurlyeq i\}$
      set $n_i = n_i + 1$.
    end if
    if $n_i = O$ **then**
      $P_1 = P_1 \cup (i)$ where P_1 denotes the first non-dominated front.
    end if
    set front count $K = 1$.
  end for
  while $P_k \neq \phi$ **do**
    initialize $Q = \phi$ for storing next non-dominated solutions.
    for $\forall i \in P_k$ *and* $\forall j \in S_i$ **do**
      Update $n_j = n_j - 1$
      if $n_j = 0$ **then**
        set$Q = Q \cup j$
      end if
    end for
    Set $K = K + 1$ and $P_k = \phi$
  end while

Algorithm 2: NSGA-II( ).

Let fronts be denoted as $F_1, F_2 \cdots F_R$.
Let objective functions be denoted as $M_1, M_2 \cdots M_K$.
Let solutions in a front be denoted as $S_1, S_2 \cdots S_i$.
$\left|F_j\right| = l$ denotes number of solutions in a front.
cd_k denotes the crowding distance w.r.t Kth objective function.
$X_{[i,k]}$ represents ith solution in the sorted list w.r.t K.
  for *every fornt* $j = 1 \cdots R$ **do**
    for *every objective function* $M_1, M_2, \ldots, M_k$ **do**
      sort the solution in F_j in descending order.
      Assign $cd_K(x_{[1\cdot k]}) = cd_K(x_{[i\cdot k]}) = \infty$
      for $i = 2$ *to* l **do**
        assign $cd_k(x_{[i,k]}) = \dfrac{z_k(x_{[i+1,k]}) - z_k(x_{[i-1,k]})}{z_k^{\max} - z_k^{\min}}$
      end for
    end for
  end for
  Total crowding distance of a solution $cd(x) = \sum_k cd_k(x)$
  i.e sum of the crowding distances with respect to every objective.

Algorithm 3: Crowding distance( ).

Let r_i denotes rank of solution i and r_j denotes rank of solution j.
  if $r_i < r_j$ **then**
    select solution i.
  else $\{r_i = r_j\}$
    select solution i if $crowd_{dis(i)} > crowd_{dis(j)}$.
  end if

Algorithm 4: Crowded tournament selection ( ).

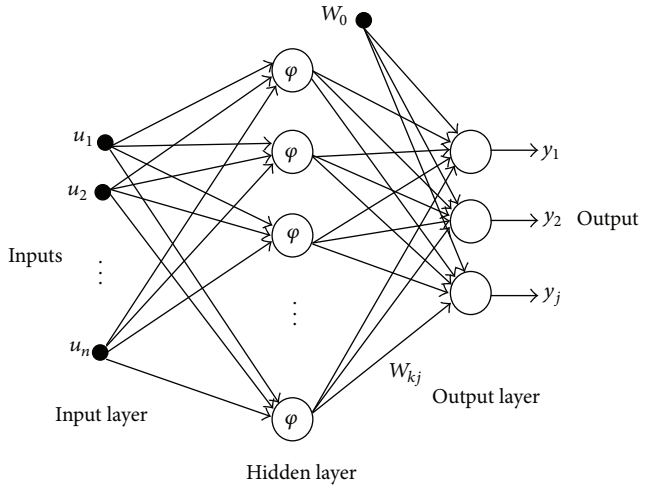

Figure 3: Architecture of radial basis function network.

and K-means clustering with HLA. HLA mostly adjusts the centres and the widths. In [35], multiobjective PSO (MOPSO) is implemented to simultaneously optimize the architecture and the connection weights of RBFN. The RBF networks are encoded as particles in PSO, and the particle evolves towards pareto optimal front to solve the trade-off problem between model accuracy and complexity. In 2010, a two level learning method for designing an optimal RBFN using adaptive velocity update relaxation PSO (AVURPSO) and OLS is implemented in [36].

Besides evolutionary techniques, orthogonal decomposition method is used to solve the least square problem to determine the update weight matrix connecting the hidden and output layer. In 1991, Chen et al., in [37, 38], used orthogonal least square algorithm for the construction of RBFN, and the forward regression procedure provided a systematic approach for supervised selection of centres. In [39], besides forward center selection, backward centre selection is proposed on batch OLS where the centres are sequentially removed while minimizing the network output error. In [40, 41], Yu et al. have demonstrated recursive version in contrast to batch OLS. The recursive version requires less computer memory and also maintains robust property. ROLSA solves to find the output layer weights, and the approach is extended to select the minimum number of centres of RBFN. The approach of ROLSA was validated on two applications such as nonlinear time series and a real multiinput and multi-output (MIMO) chemical process.

4.2. Training of RBFN Using ROLSA (Pseudocode). The backward center selection algorithm as proposed in [19] by Altschul et al. is implemented here to optimize the structure of RBFN. The pseudocode of the implementation is described below. The final updated weight matrix connecting the hidden and the output layer gave high performance accuracy when the network is used as a classifier for protein superfamily classification problem. As the orthogonal decomposition is numerically robust for solving least square problem; the final network so obtained is highly reliable with small architecture.

Table 1: Details of training and test set.

Protein family	Training sequence	Test sequence
Globin (840)	504	336
Kinase (542)	325	217
Ribitol dehydrogenase (828)	497	331
Ligase (678)	407	271

5. Experiment Details

5.1. Input Details. The four superfamilies considered in our experiment are globin (840), kinase (542), ribitol dehydrogenase (828), and ligase (678) from Uniprot database ⟨http://www.uniprot.org/⟩. From each family, 60% of total data set formed the training set and the remaining 40% formed the test set. The redundant samples were eliminated during data preprocessing stage and the data were then normalized using min-max normalization equation as given below (Algorithm 5, Table 1):

$$X_{\text{norm}(i,j)} = \frac{\left(X_{(i,j)} - \text{col}_{\min}\right)}{\left(\text{col}_{\max} - \text{col}_{\min}\right)}. \tag{10}$$

5.2. Architecture of PNN as Inductive Algorithm. The number of nodes of the input layer of PNN was fixed based on the reduced feature matrix obtained from every chromosome. The pattern layer in the PNN architecture was built using fifteen samples from every class to estimate the Gaussian PDF (probability density function) value for every test sample. The summation layer has four nodes each representing an individual class. The single output node of the PNN evaluates the maximum Gaussian PDF estimate to classify a test pattern. 50 randomly chosen samples from every class formed the test matrix.

5.3. Parameter Details of PCA-NSGA-II

5.3.1. Initialization of Chromosome. The population was initialized by encoding the chromosomes as strings of 0's and 1's. The chromosome length was fixed on the basis of number of eigenvectors having nonzero eigenvalues. In our experiment, the chromosome string length was fixed at 63. Every one bit of chromosome represents selection of the eigenvector, and zero represents discard of the eigenvector.

The population size was fixed at $N = 40$,
probability of Crossover ($P_c = 0.7$),
probability of mutation ($P_m = 0.005$).

5.3.2. Evaluation of Fitness Function. The first objective function (f_1) is to minimize the number of eigenvectors.

f_1 is computed as number of 1's in the chromosome string. The second objective function (f_2) is to minimize the misclassification error by implementing PNN as classifier, which is the measure of number of misclassified samples with respect to total number of samples.

Let **N** = number of training samples
n_h = number of randomly chosen hidden centres.
Y = desired output matrix of size $N \times p$ where p is the number of nodes in the output layer.
$\widehat{Y}$ = neural network output matrix of size $N \times p$.
ϕ = hidden layer output matrix of size $N \times n_h$
E = error matrix of size $N \times p$.
W_{hp} = connecting weight between the hidden and output layer of size $n_h \times p$.

Step 1. Perform QR decomposition of ϕ matrix.

Step 2. Evaluate $Q^T Y = \begin{bmatrix} \widehat{Y} \\ \widetilde{Y} \end{bmatrix}$ where $\widehat{Y}$ is of size $n_h \times p$ and $\widetilde{y}$ is of size $(N - n_h) \times p$.

Step 3. Evaluate the loss function $(V) = \left\| \widetilde{Y}(N) \right\|_F^2 / (N)$

Step 4. Evaluate the Akaike's final prediction error $(FPEV) = \dfrac{1 + \beta(n_h/N)}{1 - \beta(n_h/N)} V$

Step 5. Remove each network center k and compute the loss functions.

```
remove − hidden()
{
for every hidden node i = 1 to n_h do
    R(:, i) = []
    [Q'R'] = qr(R)
    OP = Q'Ŷ
    Ŷ_j = op(1 : n_h − 1, :)
    ỹ_j = op(n_h, :)
    ResK = ‖ỹ_j‖_F^2 + V/(N)
    ResCK(1, i) = ResK
end for
[minval, minind] = min(ResCK)
FPEK = FPE ∗ minval
}
if (FPEk < FPEV) then
    R(:, minind) = []
    Ŷ(minind, :) = []
    V = minval
    FPEK = FPEV
    n_h = n_h − 1
    call remove − hidden()
else
   Compute the final optimal weight matrix from the equation R_j ∗ W_j = Ŷ_j
end if
```

Algorithm 5: ROLSA( ).

This is calculated as

$$f_2 = \left(\frac{\text{No. of misclassified samples}}{200} \right). \tag{11}$$

5.3.3. Stopping Criteria. The PCA-NSGA-II process terminates when 90% or more number of solutions are in the best nondomination level.

5.4. Parameter Details of RBFN-ROLSA. From the number of solutions obtained from the first pareto front (F_1), twenty-nine (29) eigenvectors showing a misclassification error of 0.57 are selected. Twenty five (25) randomly chosen samples from the training set of every class were selected as hidden centres in the hidden layer. The output layer consists of four nodes, each representing a class.

5.5. The Parameters Used for Measuring the Efficiency of Classifier. For any classification problem, the outcomes of the data are always labelled, that is, either positive (p) or negative (n). Based on the two outcomes there may be various combinations of outputs. If the outcome from a classifier is p and the actual outcome value is also p, then it is called true positive, but if the classifier output is p and the actual outcome is n, then it is false positive. Conversely, if the actual output and the classifier are both n then it is called true negative, and if the classifier output is n and the actual value is p we refer to it as false negative. The measures used by most of the researchers are as follows:

(1) precision = $[(\text{TP} + \text{TN})/(\text{TP} + \text{FP} + \text{TN} + \text{FN})] * 100\%$,

(2) sensitivity = $(\text{TP}/(\text{TP} + \text{FN})) * 100\%$,

(3) specificity = $(\text{TN}/(\text{TN} + \text{FP})) * 100\%$,

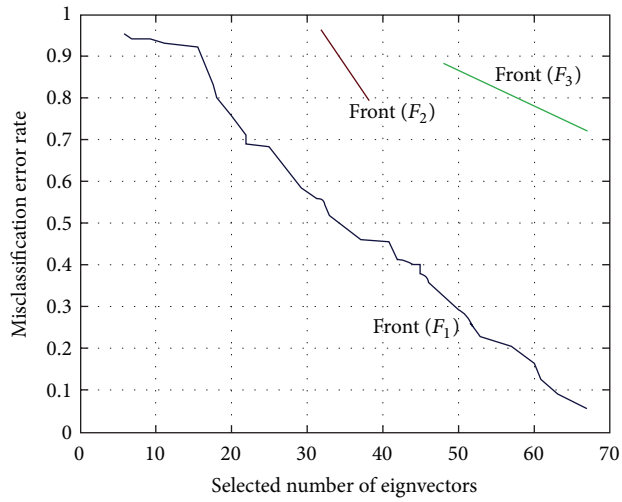

Figure 4: Graph showing pareto fronts generated after the convergence of PCA-NSGA-II.

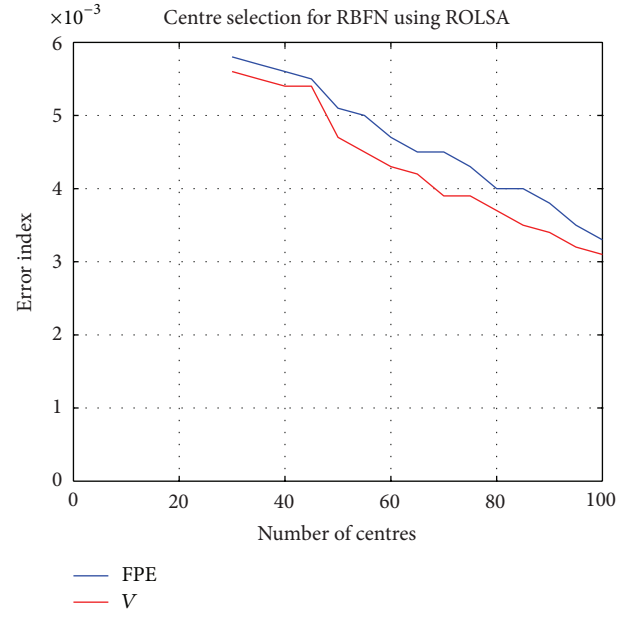

Figure 5: Graph showing Error Index versus No. of Centers.

where TP = number of true positive, TN = number of true negative, FP = number of false positive, and FN = number of false negative.

6. Results and Discussion

6.1. Results from 1st Stage. PCA-NSGA-II procedure took 2041 number of generations to meet the stopping criteria thereby generating three pareto fronts (F_1, F_2, F_3) shown in Figure 4. Few solutions in the upper, lower and mid region of F_1 are shown in Table 2.

Table 2: Pareto optimal solutions of first Pareto front.

Sl. No.	F_1 (number of eigen vectors)	F_2 (misclassification error)
1	6	0.93
2	16	0.85
3	26	0.64
4	*29*	*0.57*
5	57	0.21
6	63	0.08
7	66	0.06

6.2. Results from 2nd Stage. The 29 selected eigenvectors (from stage 1) showing a misclassification error rate of 0.57 are selected from the mid region of F_1 which formed the reduced feature matrix. The implementation of backward center selection algorithm (ROLSA) on $29 \times 100 \times 4$ network finally yielded in $29 \times 45 \times 4$ reduced architecture. The final optimal weight matrix of size 45×4 on the reduced network shows phenomenal performance in accuracy when tested on the test data set. The loss function (V), final prediction error (FPE) versus number of centres is shown in Figure 5. The number of correctly classified samples with respect to individual superfamily is shown in Table 3.

n-Fold Cross-Validation Technique. To measure the overall performance of the classifier, 10-fold cross-validation technique is implemented. The training set T is randomly partitioned into ten equal disjoint sets: $T_1, T_2, \ldots, T_{10}$. The ten RBFN classifiers are trained on the complement $\overline{T}_i$, and each classifier is then tested on the corresponding unseen test set T_i. The final cross-validation recognition rate (R) is given by

$$R = \frac{1}{10}\sum_{i=1}^{10} r\left(T_i, \overline{T}_i\right), \tag{12}$$

where $r(T_i; \overline{T}_i)$ is the recognition rate on T_i using the RBFN classifier trained on $\overline{T}_i$ [15, 18].

To show the efficiency of RBFN-ROLSA over standard neural networks, the comparison process is further extended by implementing some standard neural networks. Feedforward neural network (trained using backpropagation (BP) algorithm), PNN, and standard RBFN (trained using supervised gradient descent learning algorithm [34]) are implemented. The learning rate (η) and momentum (α) are the two main control parameters of BP algorithm. Keeping $\alpha = 0.3$ fixed and varying η in the range of 0.1 to 1, variation in performance accuracies was observed. Similarly, the smoothing parameter (σ) is the controlling parameter for PNN and RBFN which was varied to derive various performance accuracies. In RBFN-MOGA, after deriving the optimal ensemble of RBFN from the pareto optimal set, various accuracies were observed by varying σ values in the range of 0.1 to 1. The highest possible accuracies obtained by the networks are shown in Table 4 and Figure 6.

Table 3: Sensitivity and specificity with respect to every class.

Superfamily	No. of corr. classif.	Sensitivity in %	Specificity in %
Globin (840)	330	98.21%	98.77%
Kinase (542)	214	98.62%	98.61%
Rib.dehydro. (828)	327	98.79%	98.54%
Ligase (678)	268	98.89%	98.52%

Table 4: Maximum performance accuracy achieved by neural networks.

Sl. No.	Neural networks	Performance accuracy (in %)
1	FFN-BP	85.33
2	PNN	92.67
3	Standard-RBFN	84.13
4	RBFN-MOGA	96.18
5	RBFN-ROLSA	98.61

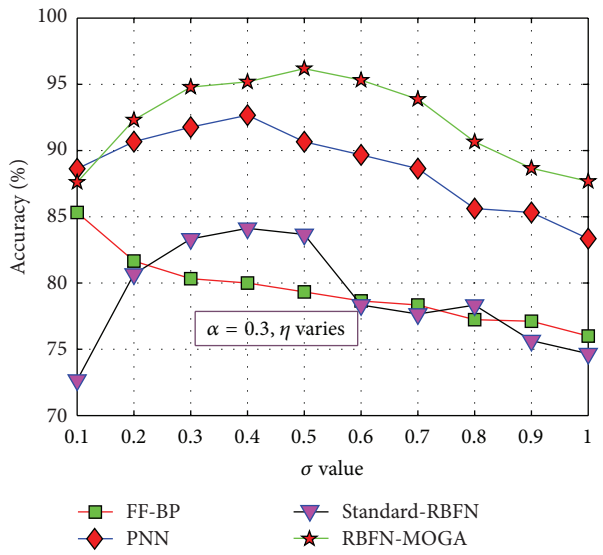

Figure 6: Graph showing performance of standard neural networks.

7. Conclusion and Scope of Future Work

From the above implementation of two-stage approach, it can be concluded that PCA-NSGA-II gave the most optimal number of significant features which was verified on PNN used as induction learning algorithm. The distinguishing features were derived from the pareto front which was the input to the RBFN. The backward center selection algorithm gave the most parsimonious structure having a very high performance accuracy value of **98.61%** which is the mean recognition rate obtained from the 10-fold cross-validation technique. Generally, the standard neural networks (FFNN and RBFN) initialize the connecting weight matrix randomly due to which their performance varies when the algorithm is implemented number of times. This approach of structure optimization using ROLSA is very reliable in comparison to randomized evolutionary techniques. The RBFN so obtained is highly robust which works efficiently on large training and test data set. Many pattern classification problems can be efficiently solved by implementing RBFN with ROLSA as a classifier. Thus, the two-stage approach has efficiently minimized number of features in the first stage and also minimized number of hidden centres (of RBFN) in the second stage thereby updating the connecting weight matrix between the hidden and output layer.

References

[1] K. Blekas, D. I. Fotiadis, and A. Likas, "Motif-based protein sequence classification using neural networks," *Journal of Computational Biology*, vol. 12, no. 1, pp. 64–82, 2005.

[2] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison, "Markov chain and hidden Markov model," in *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*, chapter 3, pp. 47–65, Cambridge University Press, 1998.

[3] C. Wu, M. Berry, S. Shivakumar, and J. McLarty, "Neural networks for full-scale protein sequence classification: sequence encoding with singular value decomposition," *Machine Learning*, vol. 21, no. 1-2, pp. 177–193, 1995.

[4] A. Edgardo, FerrAn, P. Ferrara, and B. Pflugfelder, "Protein classification using neural networks," in *Proceedings of the International Conference on Intelligent Systems for Molecular Biology*, vol. 1, pp. 127–135, 1993.

[5] J. T. L. Wang, Q. Ma, D. Shasha, and C. H. Wu, "New techniques for extracting features from protein sequences," *IBM Systems Journal*, vol. 40, no. 2, pp. 426–441, 2001.

[6] D. Wang, N. Kion Lee, and T. S. Dillon, "Extraction and optimization of fuzzy protein sequences classification rules using GRBF neural networks," *Neural Information Processing-Letters and Reviews*, vol. 1, no. 1, pp. 53–59, 2003.

[7] E. G. Mansoori, M. J. Zolghadri, S. D. Katebi, H. Mohabatkar, R. Boostani, and M. H. Sadreddini, "Generating fuzzy rules for protein classification," *Iranian Journal of Fuzzy Systems*, vol. 5, no. 2, pp. 21–33, 2008.

[8] L. French, A. Ngom, and L. Rueda, "Fast protein superfamily classification using principal component null space analysis," vol. Advances in Artificial Intelligence3501, pp. 158–169, 2005.

[9] D. Wang and G. B. Huang, "Protein sequence classification using extreme learning machine," in *Proceedings of theInternational Joint Conference on Neural Networks (IJCNN '05)*, vol. 3, pp. 1406–1411, 2005.

[10] X.-M. Zhao, J.-X. Du, and H.-Q. Wang, "A new technique for extracting features from protein sequences," in *Proceedings of the International conference on Intelligent Computing*, pp. 1223–1232, August 2005.

[11] E. G. Mansoori, M. J. Zolghadri, and S. D. Katebi, "Protein superfamily classification using fuzzy rule-based classifier," *IEEE Transactions on Nanobioscience*, vol. 8, no. 1, pp. 92–99, 2009.

[12] Z. Zainuddin and M. Kumar, "Radial basis function neural networks in protein sequence classification," *Malaysian Journal of Mathematical Sciences*, vol. 2, no. 2, pp. 195–204, 2008.

[13] S. Bandyopadhyay, "An efficient technique for superfamily classification of amino acid sequences: feature extraction, fuzzy clustering and prototype selection," *Fuzzy Sets and Systems*, vol. 152, no. 1, pp. 5–16, 2005.

[14] X.-M. Zhao, D.-S. Huang, and Y.-M. Cheung, "A novel hybrid GA/RBFNN technique for protein sequences classification," *Protein and Peptide Letters*, vol. 12, no. 4, pp. 383–386, 2005.

[15] X.-M. Zhao, Y.-M. Cheung, and D.-S. Huang, "A novel approach to extracting features from motif content and protein composition for protein sequence classification," *Neural Networks*, vol. 18, no. 8, pp. 1019–1028, 2005.

[16] M. Dash and H. Liu, "Feature selection for classification," *Intelligent Data Analysis*, vol. 1, pp. 131–156, 1997.

[17] C. Wu, G. Whitson, J. McLarty, A. Ermongkonchai, and T.-C. Chang, "Protein classification artificial neural system," *Protein Science*, vol. 1, no. 5, pp. 667–677, 1992.

[18] Y. Bengio and Y. Grandvalet, "No unbiased estimator of the variance of K-fold cross-validation," *Journal of Machine Learning Research*, vol. 5, pp. 1089–1105, 2004.

[19] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of Molecular Biology*, vol. 215, no. 3, pp. 403–410, 1990.

[20] Z. Xing, J. Pei, and E. Keogh, "A brief survey on sequence classification," *ACM SIGKDD Explorations Newsletter*, vol. 12, no. 1, pp. 40–48, 2010.

[21] S. Sharma, V. Kumar, T. S. Rani, S. D. Bhavani, and S. B. Raju, "Application of neural networks for protein sequence classification," in *Proceedings of International Conference on Intelligent Sensing and Information Processing (ICISIP '04)*, pp. 325–328, January 2004.

[22] A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical pattern recognition: a review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4–37, 2000.

[23] A. Ghosh and S. Dehuri, "Evolutionary algorithms for multicriterion optimization: a survey," *International Journal of Computing and Information Sciences*, vol. 2, no. 1, pp. 38–57, 2004.

[24] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.

[25] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[26] M. Farina, K. Deb, and P. Amato, "Dynamic multiobjective optimization problems: test cases, approximations, and applications," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 5, pp. 425–442, 2004.

[27] K. Balci and V. Atalay, "PCA for gender estimation: which eigenvectors contribute?" in *Proceedings of the 16th International Conference on Pattern Recognition (ICPR'02)*, pp. 363–366, IEEE Computer Society, Washington, DC, USA, 2002.

[28] Z. Sun, G. Bebis, and R. Miller, "Object detection using feature subset selection," *Pattern Recognition*, vol. 37, no. 11, pp. 2165–2176, 2004.

[29] R. Neruda and P. Kudová, "Learning methods for radial basis function networks," *Future Generation Computer Systems*, vol. 21, no. 7, pp. 1131–1142, 2005.

[30] A. Steve Billings and G. L. Zheng, "Radial basis function network Configuration using Genetic Algorithms," *Neural Networks*, vol. 8, no. 6, pp. 877–890, 1995.

[31] L. Guo, D.-S. Huang, and W. Zhao, "Combining genetic optimisation with hybrid learning algorithm for radial basis function neural networks," *Electronics Letters*, vol. 39, no. 22, pp. 1600–1601, 2003.

[32] J. González, I. Rojas, J. Ortega, H. Pomares, J. Fernández, and A. Fco, "Multiobjective evolutionary optimization of the size, shape, and position parameters of radial basis function networks for function approximation," *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1478–1495, 2003.

[33] T. Hatanaka, N. Kondo, and K. Uosaki, "Multiobjective structure selection for radial basis function networks based on Genetic Algorithm," in *Proceedings of the Congress on Evolutionary Computation*, vol. 2, pp. 1095–1100, 2003.

[34] Z.-Q. Zhao and D.-S. Huang, "A mended hybrid learning algorithm for radial basis function neural networks to improve generalization capability," *Applied Mathematical Modelling*, vol. 31, no. 7, pp. 1271–1281, 2007.

[35] S. N. Qasem and S. M. Shamsuddin, "Generalization improvement of radial basis function network based on multi-objective particle swarm optimization," *Journal of Artificial Intelligence*, vol. 3, no. 1, pp. 1–16, 2010.

[36] M. Majid Zirkohi, M. Mihammad, Fateh, and Ali Akbarzade, "Design of Radial basis function network using adaptive particle swarm optimization and orthogonal least squares," *Journal of Software Engineering and Applications*, vol. 3, pp. 704–708, 2010.

[37] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Transactions on Neural Networks*, vol. 2, no. 2, pp. 302–309, 1991.

[38] S. Chen, P. M. Grant, and C. F. N. Cowan, "Orthogonal least-squares algorithm for training multioutput radial basis function networks," *IEE Proceedings on Radar and Signal Processing*, vol. 139, no. 6, pp. 378–384, 1992.

[39] X. Hong and S. A. Billings, "Givens rotation based fast backward elimination algorithm for RBF neural network pruning," *IEE Proceedings on Control Theory and Applications*, vol. 144, no. 5, pp. 381–384, 1997.

[40] D. L. Yu, J. B. Gomm, and D. Williams, "A recursive orthogonal least squares algorithm for training RBF networks," *Neural Processing Letters*, vol. 5, no. 3, pp. 167–176, 1997.

[41] J. B. Gomm and D. L. Yu, "Selecting radial basis function network centers with recursive orthogonal least squares training," *IEEE Transactions on Neural Networks*, vol. 11, no. 2, pp. 306–314, 2000.